Pedro Lopes Eusébio

Licenciado em
Ciências de Engenharia Eletrotécnica e de Computadores

Applicability of Multispectral Imagery for Detection of Prescribed Fires and Rekindling

Dissertação para obtenção do Grau de Mestre em

Engenharia Eletrotécnica e de Computadores

Orientador: José António Barata de Oliveira, Professor Associado,
Universidade NOVA de Lisboa

Co-orientador: Francisco Marques, Engenheiro de Investigação,
UNINOVA-CTS

FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

November, 2020

Applicability of Multispectral Imagery for Detection of Prescribed Fires and Rekindling

Este documento foi gerado utilizando o processador (pdf)LaTeX, com base no template "novathesis" [1] desenvolvido no Dep. Informática da FCT-NOVA [2].
[1] https://github.com/joaomlourenco/novathesis [2] http://www.di.fct.unl.pt

Acknowledgements

I would like to start by thanking my Co-Adviser, Francisco Marques, for all the support and guidance throughout this last year. Even with an extremely busy schedule, he always managed to give me all the help I needed to successfully finish this dissertation. I would also like to thank my Professor Advisor José António Barata for giving me the opportunity to participate in this amazing project and the Institution Faculdade de Ciências e Tecnologias - Universidade Nova de Lisboa, where I studied for these last five years.

I want to especially thank my group of co-workers and friends from FCT, who gave me the support and incentive I needed for this dissertation's redaction. After this experience, I can say I have a group of close friends that will always be there for each other, and for that I am forever grateful.

I would also want to thank my family, especially my mother, father and brother, who never stopped encouraging me and always believed in my success. They were here for me everyday and gave me the love and stability I needed in these extremely complicated times.

Lastly, I would like to thank my girlfriend for giving me the strength and ambition to finish this dissertation. She has been with me, in my best and my worst, and was my main motivation since day one. I could not have made it without her.

Abstract

Forest fires are an increasingly relevant problem nowadays with the worsening of global warming's most severe consequences. These fire occurrences, that can cause immense damage to forest ecosystems and have a great negative impact in peoples lives, begin often with rekindles. These problems can be very difficult to tackle, needing to involve a lot of people to surveil the areas at risk. A system capable of executing this surveillance protocol and alerting the fire fighting authorities of fire and possible rekindle occurrences would be extremely beneficial in these scenarios.

A system aiming to achieve this goal is being implemented, composed of an UAV equipped with a multispectral camera, capturing aerial images of these areas. This dissertation presents a fire detection model to be used in prescribed fires and rekindling situations, identifying fire instances within the captured images. It makes use of the camera's various spectral bands to highlight the areas at greatest risk and of deep learning technology to autonomously recognise these areas.

Keywords: Forest Fires, Deep Learning, Convolutional Neural Network, Computer Vision, Multispectral Sensor, Mask R-CNN

Resumo

Incêndios florestais são um problema cada vez mais relevante nos dias de hoje com o agravamento das consequências mais graves do aquecimento global. Estas ocorrências, que podem causar imensos danos aos ecossistemas florestais e ter um grande impacto negativo na vida das pessoas, são muitas vezes iniciadas por reacendimentos. Estes problemas podem ser muito difíceis de combater, necessitando de envolver muitas pessoas para vigiar as áreas de risco. Um sistema capaz de executar este protocolo de vigilância e alertar as autoridades de combate a incêndio sobre fogos e possíveis reacendimentos seria extremamente benéfico nestes cenários.

Para alcançar este objetivo, está a ser implementado um sistema composto por um UAV, equipado com uma câmera multiespectral, que irá capturar imagens aéreas dessas áreas. Esta dissertação apresenta um modelo de detecção de incêndios para ser utilizado em situações de fogos controlados e reacendimentos, identificando ocorrências de fogo nas imagens capturadas. Faz uso das várias bandas espectrais da câmera para destacar as áreas de maior risco e de tecnologia de aprendizagem automática para reconhecer essas áreas de forma autônoma.

Palavras-chave: Incêndios Florestais, Aprendizagem Automatica, CNN, Visão Computacional, Sensor Multispectral, Mask R-CNN

Contents

List of Figures

List of Tables

Acronyms

2D 2 Dimensions.

3D 3 Dimensions.

ANN Artificial Neural Network.

AP Average Precision.

CNN Convolutional Neural Network.

COCO Common Objects in Context.

FCN Fully Connected Network.

FCT Fundação para a Ciência e Tecnologia.

FN False Negative.

FP False Positive.

FPN Feature Pyramid Network.

FPS Frames Per Second.

GPU Graphics Processing Unit.

HOG Histogram of Oriented Gradients.

IoU Intersection over Union.

JSON JavaScript Object Notation.

mAP Mean Average Precision.

MIR Middle InfraRed.

NIR Near InfraRed.

ppi pixels per inch.

ReLu Rectified Linear Activation Unit.

RF Random Forest.

RGB Red Green Blue.

RICS Robotics Industrial Complex Systems.

RoI Region Of Interest.

RPN Region Proposal Network.

SVM Support Vector Machine.

TIR Thermal InfraRed.

TN True Negatives.

TP True Positive.

UAV Unmanned Aerial Vehicle.

VIA VGG Image Annotator.

Chapter 1

Introduction

Forest ecosystems are of extreme importance for our planet's health, and its conservation are a priority. This is a task that nowadays, with global warming's most severe consequences such as temperatures increase around the globe, is becoming more and more complicated to accomplish. Forest wildfires are an aggravating problem that can also endanger people's lives and are commonly intensified by undetected rekindles. After a forest fire occurrence, existent fuel can later ignite due to latent heat, sparks, or embers, and smouldering fire, usually without a flame and at lower temperatures, can be easily unnoticed and left burning, leading to possible fire rekindles[1]. These are situations that can be really hard to detect and represent a considerable portion of the causes of forest fires. According to studies carried out with data collected between 1996 and 2015, these occurrences represent more than 14% of Portugal's forest fires known causes, with some districts having 30% of their fires caused by a rekindles[2]. A system capable of monitoring these forest areas and detect possible fire outbreaks, alerting the current fire fighting operatives and preventing the fire from spreading out, can be a proper solution to help mitigating this problem.

Recently, in the last years, Unmanned Aerial Vehicles (UAVs) equipped with image acquisition equipment such as High Resolution cameras, have been of extreme importance to remote monitoring systems, mainly in the agricultural field, allowing the users to monitor crops and fields remotely [3, 4]. The system where this dissertation is inserted consists of an UAV equipped with a Micasense multispectral camera, capable of navigating in forest environments, and with the ability to visualise and identify the presence of fire and the existence of future rekindles.

This dissertation will be mainly focused on a automatic fire detection module that will receive, in real time, data from the UAV monitoring the area, and successfully detect and locate the position of possible fire outbreaks.

1.1 Outline

This document will be organised as follows:

Chapter 1 - Introduction: Explains the problem in question and the solution proposed in this dissertation.

Chapter 2 - State-of-the-Art: Displays the various systems already implemented to solve this and similar problems and explains the different technologies, mainly in the area of computer vision and deep learning, that will be used to implement the proposed solution.

Chapter 3 - Proposed Model: Describes the proposed solution covered by this dissertation.

Chapter 4 - Implementation: Contains a thorough explanation on how the proposed model was implemented and the tools used to achieve it.

Chapter 5 - Experimental Results: Displays the several results obtained with the implemented model and presents an analysis on them.

Chapter 6 - Conclusions and Future Work: Contains a conclusion on this dissertation and some proposals for work to be implemented in the future.

1.2 Problem

More conventional fire detection systems use physical sensors that, based on the chemical properties of particles in the air, alert when in the presence of smoke and fire. These can cause false alarms since they can be activated by cigarette smoke for example, and are noticeably not suited for outside detection. A different approach must be taken regarding wildfire.

The module to be implemented should be able to receive a frame collected by the UAV's camera and confirm the positive or negative existence of fire and rekindling.

There are currently demonstrated strategies and implemented systems capable of approaching this and similar problems from different perspectives such as: an Early-Warning Fire Detection System making use of a small geosynchronous telescope with minute-scale response times[5]; Fog-assisted IoT-enabled framework for early prediction and forecasting of wildfires[6]. These examples, despite their advantages, are not suited for a system where the goal is to monitor wildfires in real time, detect future reignitions while an assessment of the plantation and soil is made and fuel mass is estimated.

Focusing on the automatic fire detection module, it is important to develop a model capable of receiving images in real time from the UAV's point of view and classifying them for the presence of fire and possible rekindling.

By recurring to Machine Learning technologies it is possible to have a more generic model, that, given a certain input, provides a predicted output based on its own parameters. In this fire monitoring example, a network can be developed to, given a certain

image, provide an answer or a confidence factor to whether or not is there fire in the input image.

1.3 Proposed Solution

The system to be implemented will have as a goal the successful detection and prediction of fire outbreaks and possible rekindles in real time, helping preventing false alarms and improving fire fighting operatives time management. The system proposed will also be able to help and improve the current process of fire monitoring and extinguishing. As mentioned before, this dissertation will be primarily focused on the fire detection and localisation module, using images captured by a multispectral camera as the main source of data, and Convolutional Neural Networks (CNNs) to implement the proposed solution. The model, after receiving a frame from the collected UAV's point of view, detects the existence of objects classified as fire and presents the user the location of every instance within that image.

For this purpose, this dissertation will make use and be focused mainly on prescribed fires to implement and present its conclusions. Prescribed fires are a specific type of fire, and created intentionally to help in forest management or ecosystems' restoration for example. This type of fire, due to its higher level of security and accessibility (explained in more detail in 4.2.1) will be the one used as main source of data. As such, the detection of common fire outbreaks that are not considered rekindle occurrences will be addressed as prescribed fires detection.

Chapter 2

State-of-the-Art

For humans, it is considered a rather basic exercise to detect different objects and differentiate them from each others, but for machines this can be a relatively complex task. For the proper functioning of this system, fire must be recognised and differentiated from other objects surrounding it.

There are already some complete algorithms in this field based on image processing techniques such as, for example, Computer Vision based method for real-time fire and flame detection[7] that is based on the detection of moving pixels with pre-specified fire-colours and high frequency activity. Fire Detection Algorithm using Image Processing Techniques[8] is based on the pixels colour, particularly the red colour, and uses image processing filters to locate edges. Fast and Efficient Method for Fire Detection Using Image Processing[9] is also based in fire colour modelling and motion detection.

All of these applications were manually computed, which required a considerably amount of intensive work, and lack a higher level of generalisation, which may lead to certain difficulties when applied in much wider scenarios. These applications, in the last few years, have been proved to be not as efficient and accurate in these fields as methods resorting to Machine Learning algorithms.

A variety of machine learning algorithms and technologies will be analysed and discussed in this chapter since, as mentioned before, can be a powerful tool in the scope of this project.

This chapter is organised in the following sections:

Section 2.2 offers a bit of context regarding image acquisition and analyses distinct options of collecting this type of data. The section 2.3 introduces some commonly used classifiers in the object detection field that can be appropriate in the proposed solution's implementation. However, since Neural Networks classification have had a bigger impact

in this field in the last few years, this will be separately addressed with a little more detail in the next section.

The section 2.4 presents the main concepts regarding Convolutional Neural Networks and object detection as well as a variety of architectures already implemented. The chapter concludes with an analysis of the technologies presented in the above sections for a better understanding of how these can be included in the proposed solution.

2.1 Introduction on Object Detection Concepts

The problem and therefore, the presented solution, fits in the scope of Computer Vision, which is a promising field that aims to build autonomous systems with behaviours similar to the human visual system [10]. In order for a system to perform any kind of computer vision work, it needs to be capable of receiving visual data like 2 Dimensions (2D) images, extract enough information from it, and perform a task based on the processed information.

Interest in computer vision triggered in the 1960s with Larry Roberts research on possibilities of extracting 3 Dimensions (3D) geometrical information from 2D perspective views of blocks[11]. This was later applied in real images instead of blocks, for possible usages in the real world. However, these methods were quite inefficient sometimes, being too specific and having difficulties adapting to different objects and environments.

This drastically changed with the arrival of Artificial Neural Networks (ANNs), with the possibility of a widely diverse training and therefore, an ability of adapting itself to distinctive scenarios. More specifically, Convolutional Neural Networks have been in the last few years a major breakthrough in the field of Computer Vision.

Systems based on Computer Vision methods are already applied in many fields such as autonomous navigation [12], objects and patterns recognition, and as mentioned in 1.2, can also be applied in fire detection.

2.2 Image Acquisition

In order to apply computer vision methods in the already described problem, firstly it should be discussed what kind of data will be analysed, what kind of patterns will be studied by the later applied networks. There are several different types of image data that can be interpreted and useful for fire detection and localisation. Even though the most basic and commonly known is Red Green Blue (RGB) image data, recorded by high-quality digital cameras, other regions of the electromagnetic spectrum like Near InfraRed for example, may be very useful and even mandatory in these fields. It is important to have some knowledge about the electromagnetic spectrum when addressing image data acquisition methods to be applied in a fire detection system, mainly the different types of waves (more specifically visible and InfraRed) and how these are measured. The visible light section on the electromagnetic spectrum, with wavelengths between 380nm and

750nm is measured by the reflection of solar radiation in the object under consideration. The Near InfraRed (NIR) and Middle InfraRed (MIR), in spite of having bigger wavelengths than the visible spectrum, are still measured in the same way, contrarily to the Thermal InfraRed (TIR), that are measured by the emitted radiation by the object under consideration, which to humans is sensed as temperature[13].

2.2.1 High Resolution Camera Acquisition

High resolution images, captured by high resolution cameras, are capable of retaining more information, compared with lower resolutions, due to its higher quantity of pixels per inch (ppi). With more pixels, these images possess a greater amount of data and can introduce additional detail that may present itself relevant to some computer vision tasks. This type of image acquisition is relatively common in object detection as it can be seen from the examples [14–16].

However, it is important to mention that, despite the common usage of these cameras in object detection, the images acquired are usually scaled down to a lower and fixed resolution. Smaller images are usually a better option in classification since their reduced amount of data enables faster and therefore more efficient training times, without losing useful information.

2.2.2 Multispectral Camera Acquisition

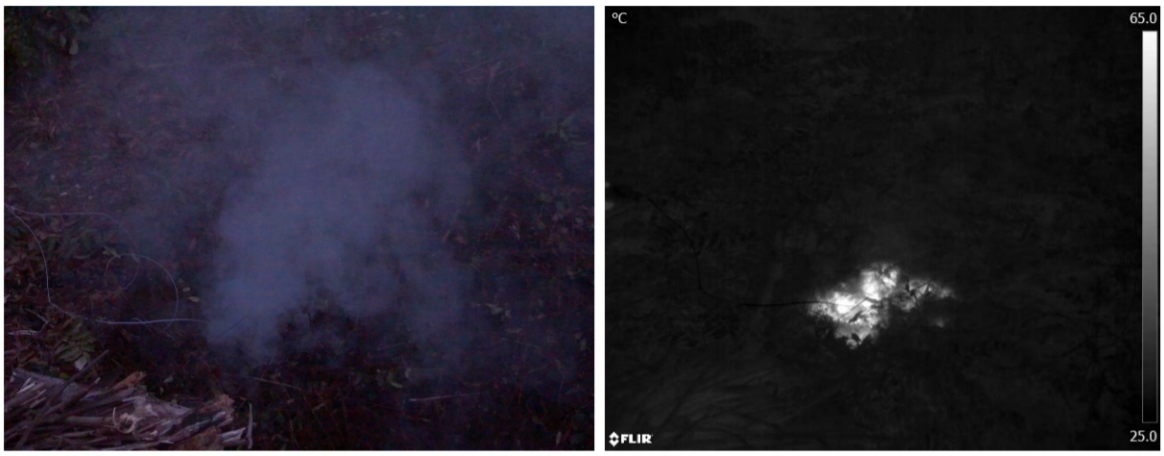
A Multispectral camera is able, in addition to collecting red, green and blue wavelengths of light, to collect wavelengths that fall outside the visible spectrum[17] such as NIR, MIR and even TIR. It usually contains between 3 and 5 spectral bands of over 100nm and its data is mainly based on the comparison between light energy reflection of objects, and surrounding objects. Despite this, there are two distinct kinds of Multispectral sensors[18]:

- **Modified Sensors:** These are standard visual sensors applied with special filters and with the ability to collect 3 spectral bands simultaneously. These filters allow various combinations of spectral bands, making it possible to capture data not only on the RGB bands but also RG-NIR for instance.

- **Multi-Band Sensors:** In contrast with the Modified sensors, the Multi-Band sensors are designed precisely for Multispectral data, containing one sensor for each spectral band. By collecting all the bands' data at the same time, the various combinations are attainable without the need to change filters and lose part of the data.

Even though a Multispectral sensor is able to recover RGB data, it is important to refer that visible light, in some cases, may not be the optimum medium with which to detect fire since visible flames can be obscured by excessive amounts of smoke. RGB imaging,

just like the human visualisation system [19] can be sensitive to visible light and be deceiving due to variations in light conditions, such as reflections. However, there is still a fair amount of implemented work in the fire detection field with the use of RGB imaging [7, 8]. In figure 2.1 there is an example showing the advantages that multispectral images can provide to fire detection, by comparing the same scenario in an RGB and TIR image.



(a) RGB image of buried fire (b) TIR image of the same buried fire

Figure 2.1: Comparison between a smouldering fire example in a RGB image in 2.1a and TIR image in 2.1b.Adapted from[20].

Since thermal imagery is considered ideal for the detection of sources of extreme temperatures, there is already a certain variety of approaches to fire detection and monitoring, making use of NIR and TIR spectral bands. Some examples are: Hybrid Contextual wildland fire detection in thermal images [21], NIR modified sensor for fire detection and monitoring [19] and classification of potential fire outbreaks based on thermal images [22].

Multispectral images are also currently used in many other fields, some of which are Agriculture[23], for a more effective crop management for instance, and Medicine[24], to evaluate certain body parts by making use of specific wavelength's properties.

2.2.3 Hyperspectral Camera Acquisition

A Hyperspectral sensor is similar to the Multispectral, but it includes a larger number of narrower bands (it can have more than a thousand bands), enabling a bigger spectral precision. These bands are usually ranged between 10 and 20nm and do not have specific names in contrast with the Multispectral ones. These sensors accuracy is very advantageous by comparison with a Multispectral sensor. However, its enormous number of different spectral bands so close to each other can generate a certain amount of redundancy and may become a considerable disadvantage in the subsequent data analysis.

Hyperspectral images, like the Multispectral, have also many diverse applications such as, for example, mineral exploration[25], due to its contribution on the evaluation

of physical-chemical conditions and geochemical evolution of hydrothermal fluids, and forensic science[26] because of its minimum risk of contamination and destruction of traces.

Due to their unique spectral signatures, Hyperspectral sensors are also used to classify vegetation and detect target species, being really useful in the forest monitoring field [27]. They have also specifically proved utility in fire related applications, showing capacity of retrieving valuable sensory data to fuel estimation, and fire severity and ecosystem recovery in the post-fire phase[28].

2.3 Machine Learning Classifiers

Classification in Machine Learning, as implied by its name, is the act a computer performs when assigning class labels to input instances. A system, in order to have a model able to perform this task, learns through an input data set that ideally, is well balanced and can represent real world scenarios[29].

For the problem in question, as already mentioned in 1.3, the objective will be a model able to classify the existence of fire in a given image. There is a wide variety of classifiers, each with their own advantages and disadvantages, that are more suited for certain types of classification problems. In this section, some Machine Learning Classifiers frequently used in object detection will be presented, such as Random Forest (RF), Support Vector Machines (SVMs) and ANNs. However this last one, given its importance in the field and its greater complexity, will be presented in more detail in the next section.

2.3.1 Random Forest

Before an in-depth analysis of this classifier, the concept of Decision Tree should be explained. A Decision Tree is a classification tool with a tree-shaped structure and different nodes performing specific tests on specific attributes. The executed tests have as outputs one or more branches and each of the tree's final nodes are associated with a class label. Following this structure, an instance starts its classification at the root node and after the specific attribute is tested, the instance follows the resulting branch to a next node and repeats this process until a class is assigned in a final node.

The RF classifier is composed of a large set of decision trees and it is mainly based on the concept that, through a bigger number of uncorrelated models working for the same goal, the overall model will become more robust and surpass its individual components (decision trees). Each individual tree predicts a class for a given input and all the output classes are compared, with the RF classifier's prediction being the class with the highest number of occurrences. By having a substantial number of trees, even when some have poor results, the majority will succeed and consequentially allow this classifier to follow the correct direction and deliver the appropriate prediction. However, for these decision trees to have some distinctive trait that results in different predicted classes, they can't

be entirely correlated. In order to achieve these individual characteristics, usually, the training set of each tree is lightly modified with random replacements from the same set, and the test's possible features in each tree's nodes are also randomised, resulting all together in greater diversity[30].An example of this classifier is presented in figure 2.2.

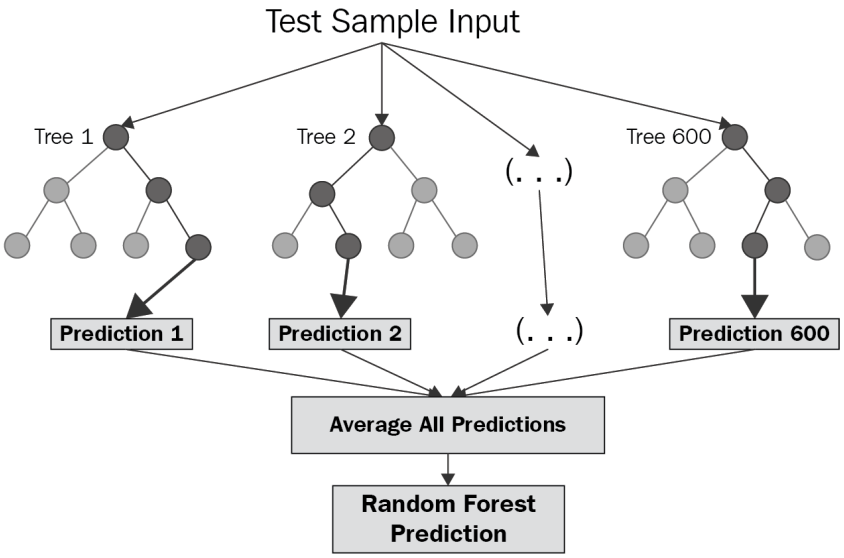


Figure 2.2: Random Forest example[31].

This method can perform in parallel machines enabling it to achieve lower training times, it is very versatile with the tasks it can perform and has also a high success rate even when in presence of an unbalanced data set. However, it does not perform as well with larger data sets, requiring a lot of memory, and it can also easily overfit, which can be considered as a huge disadvantage[32]. This classifier is relatively common in the field of image classification and even fire classification, being used for example, in a fire detection system for fire surveillance in various environments[33] and in a multispectral ensemble for forest disturbance detection[34].

2.3.2 Support Vector Machines

SVM is a type of data classification algorithm that classifies an instance based on Hyperplanes localisation. An Hyperplane (N-1 dimensions) is a decision boundary that defines in an N-dimensional space whether a data point belongs to one class or another, depending on which side of this plane the example is. In order to achieve the best plane for the given data set, it is necessary to have the maximum distance (margin) between the plane and the support vectors. These data points that are closer to the Hyperplane are the ones that help define its alignment. An example is presented in figure 2.3.

However, in a real life scenario, the data set may not be linearly separable, as shown in the example, and the use of the kernel trick is required. The kernel trick is a process where a symmetric function called Kernel Function is used to map the data set into a

higher dimensional space where it is possible to attain a linear Hyperplane[35]. In figure 2.4 can be seen an example of the kernel trick process.

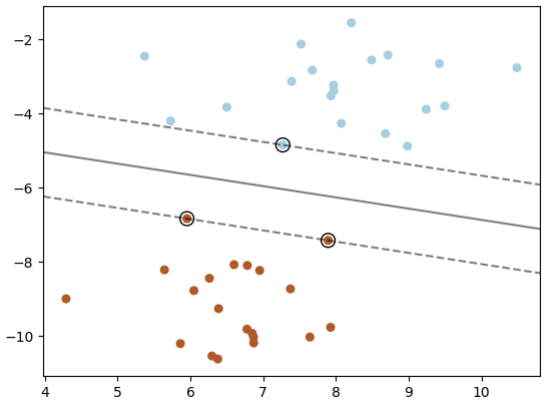
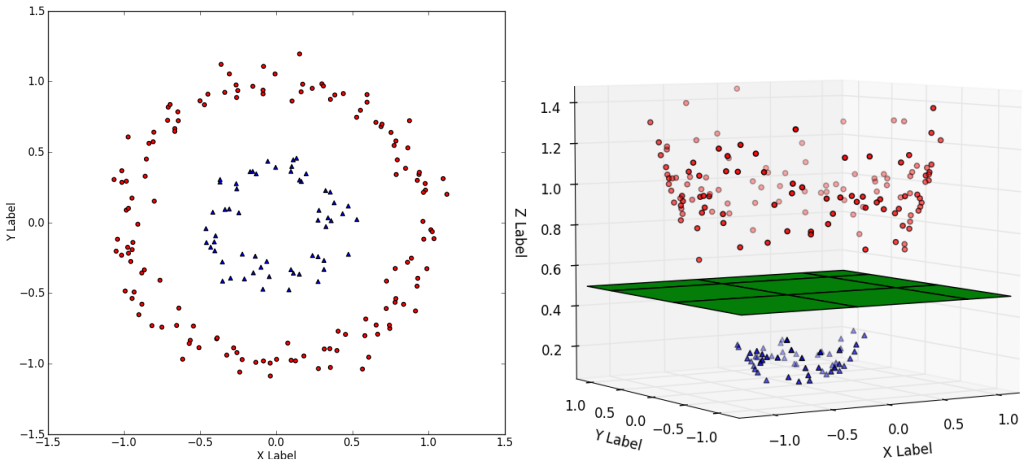


Figure 2.3: Maximum margin separating hyperplane within a two-class separable data set example[36].



(a) Non-linearly separable data in original 2D space

(b) Linearly separable data in transformed space

Figure 2.4: Kernel Trick example.Adapted from[37].

The SVM classifier is suitable for more generalised data sets, avoiding recurring overfiting, and it performs better when dealing with multi-dimensions and continuous features[29], proving itself to be an appropriate choice for image classification. Despite this, it does not perform so well with larger data sets, nor with data sets with great amounts of noise[38].

This classifier and similar variants[39] are frequently used in object recognition like for example in Selective Search for Object Recognition[40] and more specifically, fire detection in[41].

Histogram of Oriented Gradients: Histogram of Oriented Gradients[42] or HOGs can be of extreme importance for object detection scenarios and, despite not being a classifier, is often used alongside them, particularly with SVM. HOG is an algorithm arranged to detect different sections of the input image that present the possibility of having an object to classify within them. These types of algorithms are often called feature descriptors.

This particular feature descriptor is computed for each cell (the image is divided into connected regions called cells) of the image and extracts the gradient directions of the pixels, or in other words, the edge orientations, and returns an histogram with these extracted values. Each HOG descriptor is later fed to a classifier responsible of determining the presence of a given object[43].

2.4 Convolutional Neural Networks

2.4.1 Introduction on Artificial Neural Networks

Artificial Neural Networks (ANNs) are a machine learning algorithm inspired by a brain's structure capable of solving very diverse and complex problems. It is organised by multiple different layers, each composed of multiple nodes called neurons that are usually connected to other nodes from the previous and next layers. The first and last layer are the input and output layer respectively and the inner layers, since they are not visible nor reachable, are called hidden layers. Each connection between neurons has a weight that can be a negative, representing an inhibitory connection, or positive value, representing an excitatory connection, reflecting the importance of the input neuron to the output one.

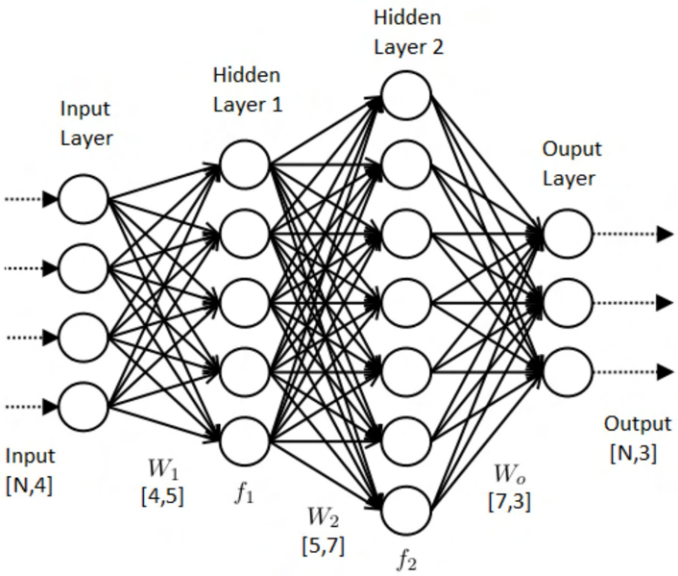


Figure 2.5: Artificial Neural Network example[44].

Despite the existence of two main groups of Neural Networks, Feed-Forward Neural

Networks and Feed-Backward Neural Networks, only the latter will be explored in this dissertation, since this category of neural networks utilises backpropagation systems to alter and rectify the networks weights in order to achieve the most accurate outputs.

There are a number of different types of ANNs, distinguished by their type of layers, and with different objectives, capable of achieving better results in specific work fields, such as for example the Recurrent Neural Network which can be very efficient in text-to-speech conversion technology. Since the proposed module inserts itself in the computer vision field, a more suitable type of ANN for object detection will be addressed, called Convolutional Neural Network.

Convolutional Neural Networks (CNNs) are a certain type of neural networks distinctive by their specific hidden layers, such as the convolutional layers. These are usually present in projects regarding computer vision and object recognition, but can also be used in other fields such as natural language processing[45]. The image data and networks input is usually represented by a tensor, that is essentially a generalised matrix, an N-dimensional data structure. CNNs were proven a few years ago to be extremely efficient in image classification and object recognition by showing higher image classification accuracy on the ImageNet Large Scale Visual Recognition Challenge[46–48].

For a better perception of how these networks work and how are they included in the scope of Artificial Neural Networks, there are some concepts that need to be understood first. This background information regarding CNNs and ANNs will be explained in the following subsections.

2.4.2 Training

Training is the action any ANN performs when it is expected to change its outputs and therefore, change its weights, which are responsible for the networks capacity of responding accordingly. To achieve weights that are suitable for the implemented network, a lot of iterations are necessary, the network needs to practice to learn from experience, and therefore, it needs a large amount of data to be its examples, the **training set**.

There are two more important data sets besides the training set, these being the validation set and the test set. The **validation set** is composed of a series of images to evaluate the network while the training is occurring. After each time the network runs over the entire training set, this set is presented to the network so the model can evaluate itself and conclude over its performance. The **test set** is also composed of a series of images to evaluate the network, but is only presented to the model after the training phase is complete. This set serves to perform a final evaluation of the model, and should be composed of unique and different images from those present in the other sets, and with all the variety of classes that the model can be presented with, when facing real world problems.

It should also be clarified what exactly is an iteration. Each time the network receives a new input from the training set, a forward pass through the network is performed and an

output with a prediction is obtained and compared with expected values. By making use of a loss function, the loss of the prediction is determined, in other words, it is generated a value that quantifies the predictions' accuracy.

After this process, Back Propagation is performed, and the networks weights are changed in accordance with the loss value. In order to obtain weights that minimise the networks loss, an optimisation algorithm is necessary. The most commonly used is Gradient descent and some of its variants. This algorithm iteratively moves in the direction of steepest descent that is defined by the negative of the gradient (presented in figure 2.6).

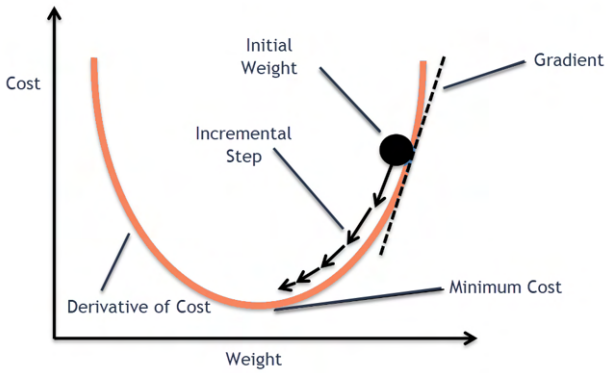


Figure 2.6: Gradient Descent visualised[49].

When the current backward pass is over, an iteration is completed and the model repeats the process for the next input image. Initially, the Neural Networks weights are random, but after each iteration performed, the network slowly changes its parameters, aiming to achieve the most accurate outputs.

When training a neural network, it is crucial to keep monitoring the test and validation results to avoid possible overfiting. Overfiting may occur when the training set lacks diversity or it is run over too many times, forcing the network to become so specific that can only acknowledge the training set and is not able to classify properly any given image outside of it. This common problem can be detected when the validation results start diverging from the training results, showing worse results than before, while the training results continue to improve. It can be prevented by penalising the model (loss function) in the training phase proportionally to its weights magnitude or by just simply reducing the complexity of the network and increasing the training set.

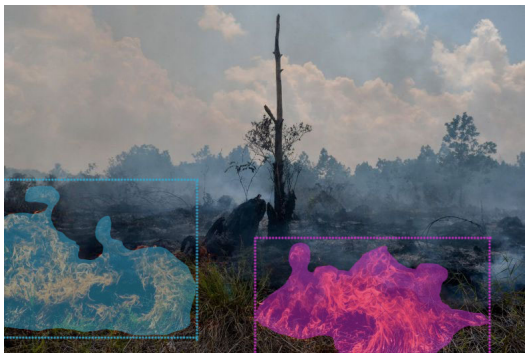
2.4.3 Classification and Segmentation

After the networks training is complete, this should be able to classify an image properly, by receiving its data and returning an array with the values corresponding to the level of confidence in the presence of an object belonging to every existent class.

There are other tasks besides classification, that can be performed by a CNN in the scope of computer vision, with different complexity levels:

(a) Classification

(b) Semantic Segmentation



(c) Instance Segmentation

Figure 2.7: Image Classification examples. Adapted from[50].

- **Semantic Segmentation:** When a network performs Semantic Segmentation, it identifies and labels all the pixels belonging to the same class, enabling the localisation of the classified object.

- **Instance Segmentation:** In instance segmentation, just like in semantic segmentation, the pixels from the same class are labelled, but if there is more than one instance from the same object, these are also taken into consideration, and labelled separately. Each instance from the same class is localised and identified as a distinct element.

It may be observed in figure 2.7 the differences between these tasks, where firstly the image is only classified as having an object of the class fire in it, secondly is is also identified where in the picture the fire pixels are located, and thirdly, it is identified the existence of different burning areas and localised these areas' pixels within the image.

2.4.4 CNNs Layers

In order to acquire a better understanding on how CNNs are structured and what differentiates them from other types of ANNs, some in-depth explanation will be presented regarding its architecture, the different behaviours of each CNN layer and the dynamics

between them. By examining the figure 2.9, it is possible to have a better overview of the CNN architecture.

2.4.4.1 Convolutional Layer

The Convolutional Layer is composed by a number of nodes, each of them being a mask called "filter", that are firstly composed of random weights. These masks will be applied to every input of the network trying to extract information from the input image, generating a Feature Map consisting of just a small part of the image's relevant information.

In the convolution process, when the filter matrix, also called kernel, is applied to a given input value, the matrix weights are multiplied by the input values corresponding to the same positions in both matrices. The sum of these multiplications is the corresponding output, and the filter proceeds to slide to the next position on the input matrix, repeating the same process to the input matrix values and thus obtaining output feature maps.

In figure 2.8 it is represented a simple example of a convolution, where the filter (middle) is being applied to the input image red value (left), resulting in the red value of the output image on the right. In this case, a one dimensional matrix is exemplified. However, in an RGB image for example, there are three channels (dimensions), which require three masks, and three feature maps are acquired.

CNNs are named after this layer since this is what better differentiates them from other types of ANNs. It also explains why CNNs are so frequently used in Computer Vision, since it detects certain types of shapes and patterns by using these filters, which can later lead to a more complex object detection. Just like in most ANNs, a greater complexity level can be obtained by using a larger amount of layers. Ideally, by adding more convolutional layers to these networks, they will be more likely to detect specific objects other than just simple lines or shapes.

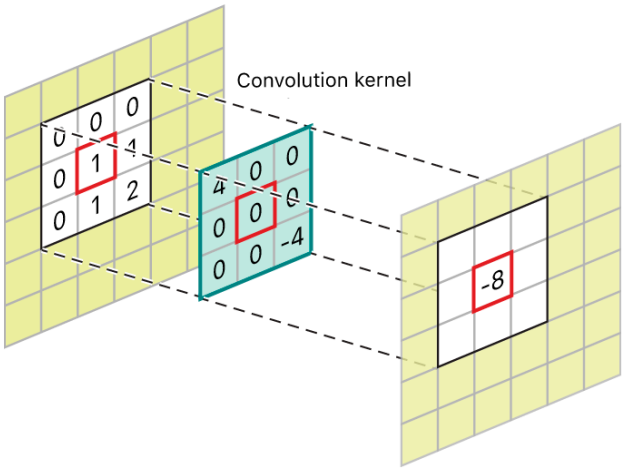


Figure 2.8: Convolution example. Adapted from[51].

2.4.4.2 ReLu Layer

The Rectified Linear Activation Unit (ReLu) is a non-linear activation function that is used instead of for example, the sigmoid function, since the data that is treated in these kind of problems is usually non-linear. This function avoids easy saturation and returns the exact pixel value if this is positive, and zero if the input is negative. Despite not being considered exactly a layer, every different feature map needs to be passed though it.

2.4.4.3 Pooling Layer

This layer is usually applied after the non linearity and its goal is to reduce the information obtained in each feature map to the most crucial extracted features. This process, called down sampling, involves the extraction of a different version of the layer's input with the same amount of valuable elements and lower resolution. This is a necessary step to be taken in a CNN, since the input image can suffer some changes regarding its positioning and consequentially have different feature maps extracted, despite these features being positional invariant.

2.4.4.4 Fully Connected Layer

Frequently the last layer in these networks, the Fully Connected layer, presents the CNN's output composed by the various values, representing the level of certainty of the given network's input to belong to each one of the computed classes. In this layer, similarly to what happens in common ANN's, the different features detected previously are received as inputs and the image is classified according to the pre-existing labels[52]. It should also be mentioned that a softmax activation function is used to produce an output value between 0 and 1.

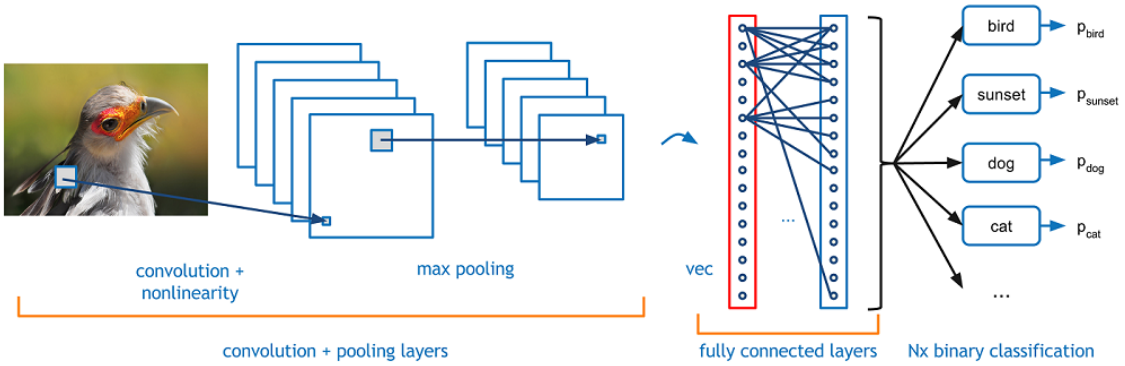


Figure 2.9: Convolutional Neural Network Structure[53].

2.4.5 Frameworks

There are a number of different architectures based on CNNs and in order to be chosen the more adequate for this problem, their main differences were studied and are explained

below:

2.4.5.1 R-CNN

One of the first object detection frameworks to be implemented, this solution emerged to prevent the unknown number of objects to detect from being a problem. It generated, through selective search, 2000 Region Of Interests (RoIs) that are afterwards converted to an RGB image, fed as input to a CNN and classified by a SVM classifier (explored in figure 2.3.2) to predict the presence of a specific object. Although this was considered a rather fast algorithm when implemented, it still took 47 seconds to classify a given image on a GPU, since the CNN's pass throughout all the object proposals is executed without sharing computation resources[46].

2.4.5.2 Fast R-CNN

Built upon previous work, Fast R-CNN tried to improve detection accuracy and its speed, achieving nine times less training speed than the regular R-CNN architecture. This was accomplished by, instead of feeding the CNN with every one of the 2000 region proposals and execute an equal number of convolutions, the image is directly analysed by the CNN, having the convolutional operation done only once. After finishing the convolutional and max pooling layers processes, the convolutional feature maps are extracted. With these, the region proposals are identified and later fed to the fully connected layer. Every RoI has two output vectors associated, one with the class label (softmax probabilities) and the other with per-class bounding-box regression offsets[54].

2.4.5.3 Faster R-CNN

In order to achieve even shorter classification and detection times, the Faster R-CNN algorithm eliminated the selective search in the process, and instead utilises another network, the Region Proposal Network (RPN) specifically to propose candidate objects' bounding boxes. This architecture has a very similar structure to Fast R-CNN's, and produces as well two different vectors associated with every region proposed. By using the networks ability to learn the region proposals and saving the time spent on the selective search process, this architecture became faster (7 Frames Per Second (FPS)) and capable of executing objects detection in real time[55].

2.4.5.4 YOLO

You Only Look Once (YOLO) algorithm, just like its name suggests, does not require the use of regions of interest to localise any object successfully. Instead, it is only applied a single convolutional network to predict object bounding boxes and their respective probabilities in one evaluation. The input image is initially split into an SxS grid, in which every cell has a fixed number of bounding boxes that, after being classified by

the CNN, have probability and offset values associated. If a bonding box has a higher class probability value than a given threshold, its object is classified. This architecture is extremely fast compared with previous architectures, managing to achieve speeds of 45 FPS [56].

2.4.5.5 SDD

The Single Shot Multibox Detector, similarly to YOLO, utilises a single network and even excels it in terms of speed and accuracy. It makes use of a default set of bounding boxes, rather than predicting them, with predicted category scores and box offsets through small convolutional filters applied to feature maps [57].

2.4.5.6 Mask R-CNN

As an extension to Faster R-CNN, the Mask R-CNNs architecture is fairly similar, but with its new primary goal being instance segmentation. As output, besides the class label and bounding-box offset for each candidate object, it is also returned a binary mask for each RoI. Firstly the object's bounding boxes are proposed by the RPN network and, while the class and box offset are obtained, the mask for each region is predicted by applying a fully convolutional network that classifies all the bounding box pixels as either object or background [58].

2.5 State-of-the-Art Observations and Conclusions

The background information and methods presented above shall be compared and discussed for better decision-making on the proposed system.

2.5.1 Image Acquisition

By using the most appropriate data acquisition method for distinguishing fire and soil characteristics, there is a greater ease in the development of the module proposed, since the network will have better and more specific patterns to recognise. It is essential to analyse spectral bands in and outside of the visible spectrum to acquire a broader set of data, capable of highlighting possibly relevant characteristics for this application. As mentioned in 2.2.2, there is already some fire related work making use of RGB, NIR, MIR and TIR, emphasising the possibilities and advantages that using these spectral bands can bring to the proposed solution.

Both Hyperspectral and Multispectral sensors can retrieve information concerning these spectral bands. Hyperspectral sensors are presented as a more meticulous version of Multispectral sensors by being able to retrieve more specific data, but their complexity, as mentioned in 2.2.3, is a big disadvantage face to Multispectral sensors.

2.5.2 Object Detection

As mentioned earlier in 2.3, some of the classifiers reviewed in this chapter, despite the importance they had in object detection, are not as relevant to this field nowadays as Convolutional Neural Networks. However, even just regarding CNNs, there are several different parameters and architectures that need to be discussed.

By comparing the various CNN architectures it is possible to detect obvious differences, mainly concerning performance and speed. Speed is a very significant aspect and, in some real time systems depending on limited timings, having detection times as short as possible can be crucial. Nevertheless, it is important to keep in mind that for fire detection, although the module classification's speed is of extreme relevance, its accuracy is of equal or even greater importance, in order to avoid miss-classifications and consequently false alarms.

The module to be developed should also be able to detect objects from more than one class as it may prove to be an advantage in the future (smoke detection for example) and detect where in the image the various classified objects are. In other words, it would be ideal to have a system capable of classifying and localising, and of performing Instance Segmentation. The Mask-RCNN method, from the ones listed above, is the only one capable of performing instance segmentation, making it the most appropriate method for fire recognition. Nevertheless, a benchmark is still displayed in section 5.1 to assure the most appropriated framework integrates the proposed solution.

Chapter 3

Proposed Model

The objective of this dissertation is to implement an automatic fire detection module with the use of machine learning algorithms to help fighting and preventing wildfires. The proposed model, as displayed in figure 3.1, is a classification system focused on presenting the user an accurate fire detection within a given image or, if intended, present the areas where rekindles are more likely to be triggered. It makes use of an object detection framework, the Mask R-CNN, which was carefully selected to perform the classification and localisation processes. In section 5.1 it is minutely presented the framework selection process and the results that support the preference for this framework in these circumstances.

This model can be divided in two major processes, prescribed fires detection process and rekindle detection process, both explained in this chapter. The prescribed fires detection process will essentially receive an image in real time from an already established spectral band, detect the presence and location of fire instances and save an image with coloured masks around burning areas. The rekindle detection process, on the other hand, will receive the same image in the thermal band and in another spectral band, detect areas at higher temperatures where rekindling is more likely to initiate and, in the end, also return an image with coloured masks around these areas. When operating with the model, the user has the option between running the first process or the second.

In fire detection and classification, when using data collected by a standard RGB camera, the amount of smoke in the air can have a negative impact in the overall performance. It can seriously block the aerial view preventing a classification model from detecting accurately the fire location within the captured images. The presented model was implemented making use of multispectral imagery which can mitigate this obstacle, with some spectral bands being mainly used for capturing sources of high temperatures. Multispectral sensors are also not as sensitive to light conditions as RGB, suffering less

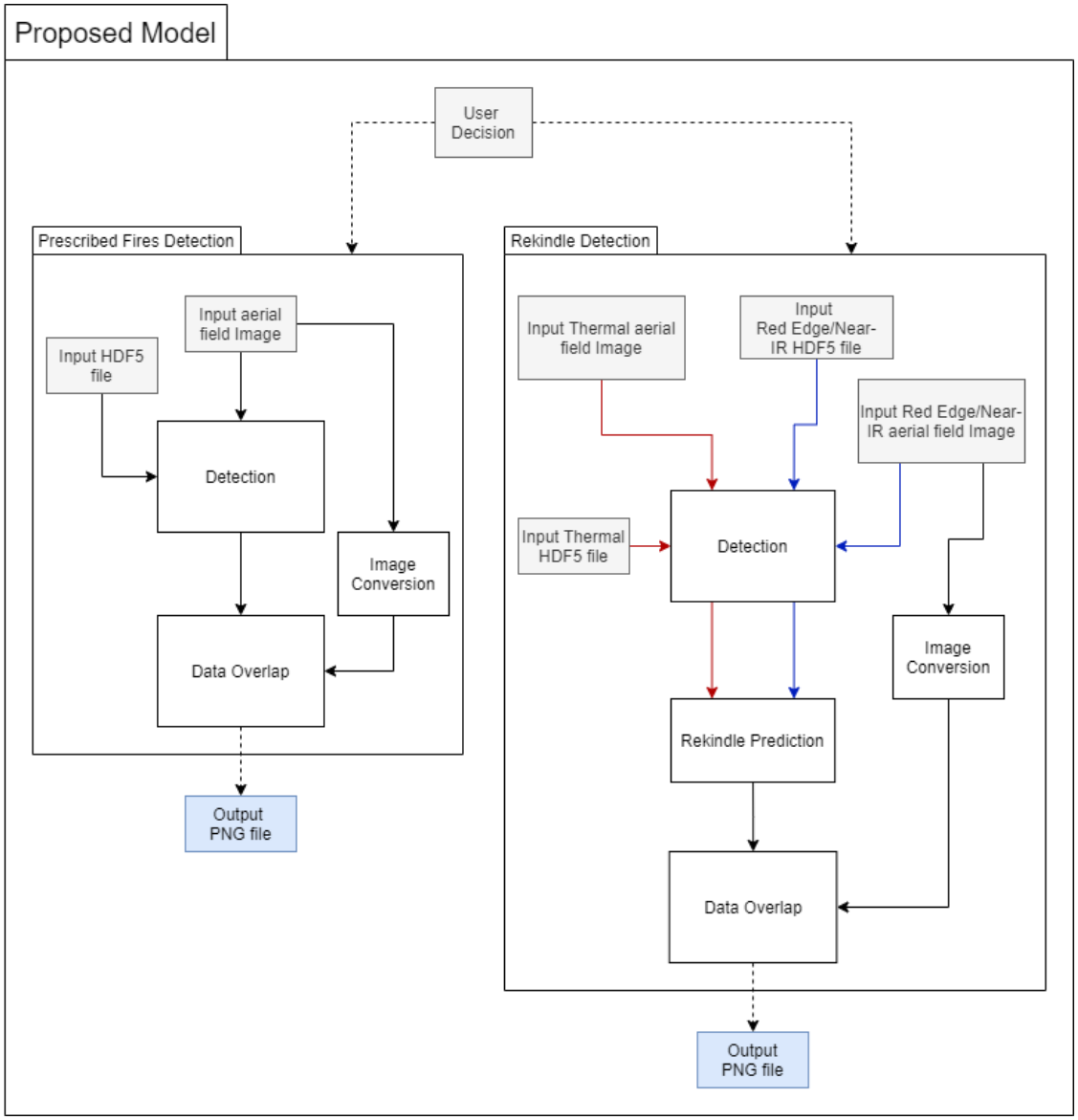


Figure 3.1: Proposed model structure, divided in two main processes, prescribed fires detection and rekindle detection.

from reflection occurrences.

An UAV equiped with a Multispectral camera with five high-resolution narrow bands integrated with a calibrated radiometric thermal sensor collects aerial multi spectral images from the field, which are used by the model as input. The proposed model performs the detection processes in images from the Red Edge, Near Infrared or Infrared (thermal) bands. These bands were considered the most appropriate and were selected to integrate in the model, mainly due to their low smoke sensitivity, unlike the RGB ones. The Red Edge and Near Infrared bands, as it can be observed later in 5.3, are able to capture the data, highlighting fire instances and ignoring almost entirely the presence of smoke that would, otherwise, block the view. The Thermal channel, on the other hand, is able to display all high temperature areas, which can be used to confirm fire occurrences and detect areas that despite not being burning, may originate rekindles in the future.

The model also needs to receive as input an HDF5 file containing the weights to the network trained with and to a specific spectral band. This weights file has a key role in these processes as it provides the network the connections values between neurons, essentially determining the networks behaviour. For example, the HDF5 file obtained by training the network with Red Edge imagery is only appropriate to detect fire in Red Edge images. To classify an image from a different band, a different weights file would be necessary. The process implemented to collect this input files is further explained in 4.2.1.

3.1 Prescribed Fires Detection

The Prescribed Fires detection process performs on a single image, receiving this and a single HDF5 file with the appropriate network's weights as inputs. This process can also be structured in two different sections, the detection section and the overlapping section. The detection section is responsible for localising possible fire occurrences within the received input image and the overlapping section for rearranging all the elements received from the previous section with the image and returning the model's final output.

In the detection section, the Mask R-CNN model runs in inference mode through the input image with the given weights and detects all pixels belonging to burning areas. It generates bounding boxes around the RoIs as well as binary masks within the bounding boxes with 1 values representing the detected object and 0 values representing anything outside that class instance. This block returns as output four arrays with data regarding the masks surrounding the detected fire areas: **rois** with detection bounding boxes, **masks** with instance binary masks, **class_ids** with class integer identifiers and **scores** with float probability scores.

Due to encoding differences between the original input images collected by the MicaSense camera and the standard accepted format in the overlapping section, the received image has to be properly adjusted. The original image loses bit depth, changes its number of channels and is then introduced as input in the next section. It occurs simultaneously

with the detection section, which makes use of the original input image, and it is only relevant to the overlapping section.

Upon receiving the adapted image, as well as the arrays, the overlapping section is responsible for arranging these elements in a way the user can understand, by overlapping them. A diagram of this process is displayed in 3.2.

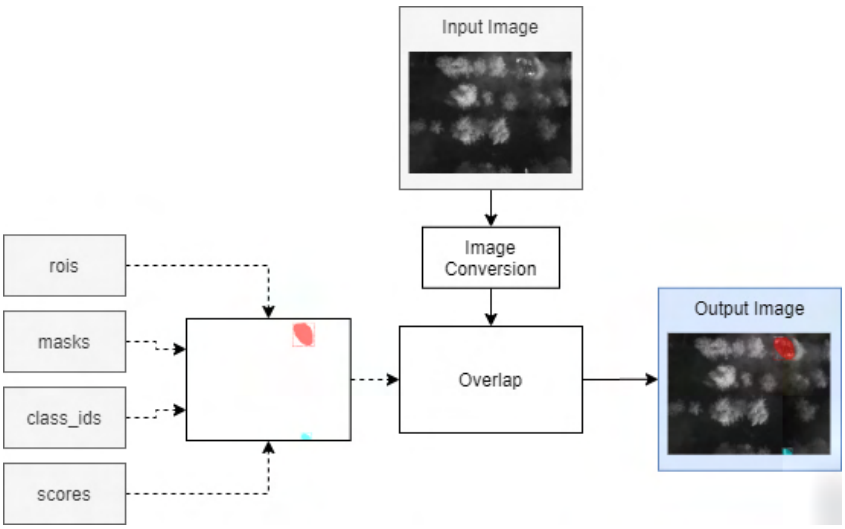


Figure 3.2: Overlapping process structure.

Having the masks and image on the same plane is an important step to enable visualisation but also crucial to, in future implementations, calculate the burning areas' actual sizes and displacements with geographic coordinates conversion. However, despite its current relevance, the visualisation step may become eventually unnecessary if this model is used, for example, to measure and study burned ground areas, where all the operations necessary can be performed after the overlapping section, without the need to display the classified image. With access to the UAV's altitude and focal length it is possible to convert the pixel sizes to meters and conclude about the detected instances' areas. This is exemplified in figure 3.3.

This section makes use of its input image and overlaps it with the previously computed masks and RoIs. It delineates in the image every RoI and within each one, assigns every pixel classified as a fire object the same random colour. It also prints next to every instance its class name, "fire", and its detection confidence value in percentage.

When this section is over, the prescribed fires detection process provides as final output a PNG file with the initial input image coloured in all the areas where fire was detected.

3.2 Rekindle Detection

The Rekindle Detection process, contrary to the Prescribed Fires Detection process, receives the same image in two spectral bands, the thermal band and the Red Edge or Near

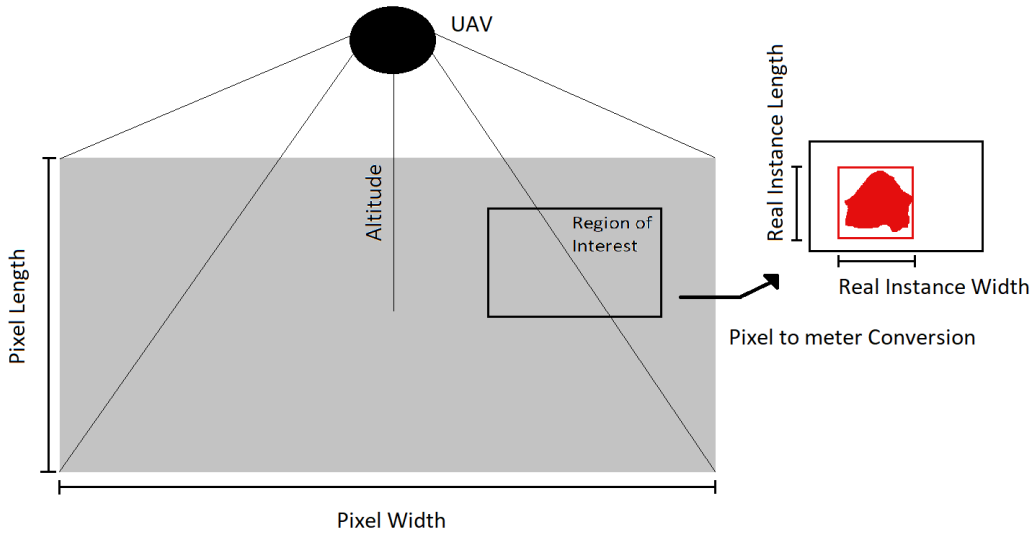


Figure 3.3: Instance Area exemplified.

Infra Red. The thermal band is necessary to collect the high temperature areas and the other band (Red Edge or Near Infra Red), as explained earlier in this chapter, is the most appropriate at displaying the burning areas locations. It also receives as input two HDF5 files with the network's weights to perform the mask detection on both spectral bands. It is divided in three sections, the same Detection and Overlapping sections like in the previous process and the Rekindle Prediction section. The Detection section is performed for both input images, one after another.

Having access to the image's thermal band is essential for an appropriate rekindle detection, since this data displays all points at extreme high temperatures. In the proposed model, when the mask detection is applied to the thermal data, all high temperature points are detected instead of just the visible burning areas. When an area with an extremely high temperature is labelled as fire in the thermal detection, it is considered an area that has been in a high temperature environment for a reasonable period of time. As such, it is also considered that all thermal detected areas have been or are still burning when the image is captured. With this data and with access to the burning areas through the other image (Red Edge or Near Infra Red) detection, it is possible to compare both and start inferring about the rekindle areas localisation.

The Rekindle Prediction section receives both detections data from the previous section and is responsible for uncovering possible rekindle areas. In this section, the areas that are considered at extreme temperatures(Thermal detection) but not considered burning(Red Edge/Near-IR detecion) are gathered. These most likely represent areas that are no longer burning but are still at high temperatures, presenting a probability of rekindling. Some of these that are too close to burning areas and will eventually be extinguished along with the other fire instances during the fire-fighting operation, do not present useful information and are excluded. Some areas with high intensities, if

too distant from the main fires and do not present visible evidences of burning, may be easily neglected. These more isolated occurrences are the main focus in this selection, being considered very likely to show signs of future rekindling. These areas are the ones selected as possible rekindle instances and their masks, class names and confidence value are the outputs of this section.

In the overlapping section, it is received as input the areas previously obtained and the same method described before in 3.1 is applied. In this case, the received image is the one that can be interpreted by the user, with the Red Edge or Near Infra Red band data. This image also needs to be adjusted, like explained earlier, before entering in the overlapping section. After merging all the information, this section returns as the model's final output the image with coloured masks around every area where a rekindle is most likely to occur.

Chapter 4

Implementation

This chapter will include and examine the implementation of the model detailed in the previous section. It is divided in three main sections, Mask R-CNN framework, Training Procedures and Detection. The Mask R-CNN framework section describes and explains how this framework which was chosen to be a fundamental part in the proposed model is structured. The training procedures section details the prior work necessary to achieve the proper conditions to perform the training process as well as the training itself. Finally, the detection section is focused on the inference process and the work implemented to integrate the Mask R-CNN framework and the training process output in the proposed model.

4.1 Mask R-CNN Framework

As mentioned in the proposed model chapter, the Mask R-CNN framework is inserted in the model to perform the desired object detection. This framework, despite its complexity, is fundamental to the implemented model performance and its basics need to be understood. This section will focus on detailing its overall practice, its structure and its most relevant features.

The Mask R-CNN was developed and released in 2017 by Waleed Abdulla, has an open source implementation and is available on this GitHub repository[59]. It is a framework with all of its code documented for easy access, implemented on Python 3, TensorFlow and Keras. TensorFlow is an open source library for machine learning created by Google Brain [60]. It has a large and flexible set of tools that allow easier building and deployment of applications based on deep learning algorithms. Keras[61], on the other hand, is a deep learning API that runs on top of TensorFlow and takes full advantage of its features. It works with python and is very user friendly, containing isolated modules that can be

incorporated into the users projects.

4.1.1 Architecture

The Mask R-CNN inference process can be divided in two main stages which are described below[58]. A diagram of this process is also displayed in 4.1.

4.1.1.1 Backbone Architecture

The first stage in Mask R-CNN detection and segmentation is the **Backbone Architecture** which is the product of a standard backbone combined with a Feature Pyramid Network (FPN). A backbone is a CNN with the primary goal of extracting features from raw images. In Mask R-CNN, both ResNet50 and ResNet101 (Residual Networks) can be used as a backbone, with the main difference distinguishing them being the number of layers. The first layers detect more basic features like shapes and edges and the last detect the more complex ones like fire and smoke for example (Bottom-Up Pathway). With FPN, these higher level features are passed to lower layers (Top-Down Pathway) and feature maps with equal spatial size from both pathways are merged (Lateral Connection)[62]. When finished, this process returns the obtained feature maps as its output.

4.1.1.2 Region Proposal Network

These feature maps are sent to be the input of the next phase, the Region Proposal Network (RPN), that has as main objective to propose candidate object bounding boxes. In this phase, a fixed size window slides over the CNN feature map, producing a set of k common aspect ratios called anchor boxes. The anchor boxes that belong to the background class and the ones with IoU smaller than 0.5 are removed. Intersection over Union (IoU), expresses the percentage of area that the predicted box and the ground truth box have intersected, with 1 representing the perfect overlapping of both boxes. In the end, the RPN outputs a bounding box per anchor, called proposed region, and a confidence score for containing an object within that region[63].

4.1.1.3 Head Architecture

With this step completed, the process enters the **Head Architecture** stage. It is very important to have a clear understanding on the differences between this stage and the Backbone architecture stage since the training of each one can be selected and have very distinct consequences to the networks results. The Head architecture is organised in two main parallel steps, one to generate classes and bounding boxes and the other to generate the masks. However, before starting these steps, the RoIs from the proposed regions need to be resized to have all the same standard fixed size. Pixel level segmentation requires an extremely precise alignment and the feature maps from each RoI need to be

perfectly extracted. This is achieved through a method called RoIAlign that uses bilinear interpolation, getting the most precise values for rounded pixels when resizing[64].

On the mask generation step, the fixed size feature maps are received into a Fully Connected Network. An array of matrices known as binary masks is returned, with the pixels containing the detected object represented by 1 and pixels with background by 0.

At the same time, for the bounding boxes and classes generation, the RoIAlign output is also received into a Fully Connected Network. It is added a softmax layer to output a classification and in parallel, a linear regression layer to output the bounding box coordinates[63].

Finally, the Mask R-CNN combines and returns the classification list mrcnn_classes and the boxes list mrcnn_bbox with the masks array, resulting in an accurate instance segmentation.

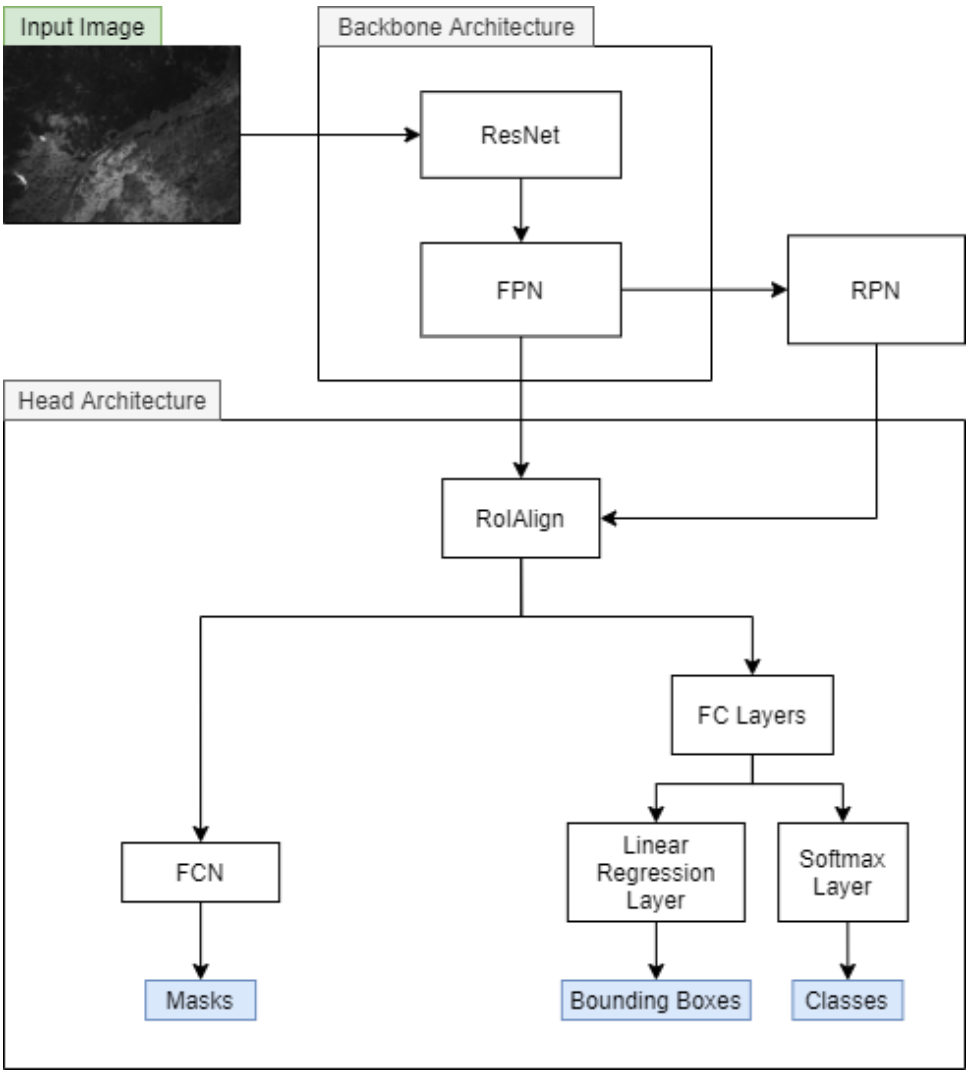


Figure 4.1: Mask R-CNN model Architecture, adapted from [58] and [63].

4.1.2 Framework Files

The Mask R-CNN framework is divided in three main folders, logs, samples and mrcnn. The **logs** folder is where the network, while training, stores its output HDF5 files. The **samples** folder consists of multiple folders, each regarding a different training data for different detection results. In this case, a fire folder was created to contain all data required to train the fire detection network. The **mrcnn** folder contains five essential python files, and only one of these was not used in the proposed model. The parallel_model.py is a file with processes designed to support Multi-GPU for Keras, which the author could not use without access to multiple GPUs. The rest of the python files are briefly described bellow:

4.1.2.1 config.py

The config.py file contains the base configurations class with all important parameters responsible for delineating the networks training and inference behaviours. These parameters are set with default values and are usually adjusted through a different python file.

4.1.2.2 model.py

This file contains the main Mask R-CNN model implementation for both training and detection. It is organised by the network layers, containing also loss values computing, data generation methods and data formatting methods. It also includes the MaskRCNN class which encapsulates the Mask R-CNN model's functionalities with, for example, the train and detect functions.

4.1.2.3 utils.py

This file consists mainly of common utility functions and classes to be used by other files' functions in the framework. It contains a great variety of methods like, for example, collecting the data from the data set's location, resizing images along with their masks and finding matches between predictions and ground truth instances.

4.1.2.4 visualize.py

The visualize.py file, as the name suggests, is the file containing the display and visualisation functions. It is a really important file since it is the one with the tools responsible for applying the obtained mask as well as displaying the output images. It includes functions for drawing the RoIs and painting the masks and bounding boxes with random colours. It also contains functions aiming to help visualize the networks performance with, for example, precision plots and statistics for the obtained weights.

The overlapping section has a crucial role in the proposed model and this file does not possess all tools to be used directly in the model. Because of this, visualize.py was the only file from the original framework that was directly modified.

4.2 Training Procedures

4.2.1 Data Gathering

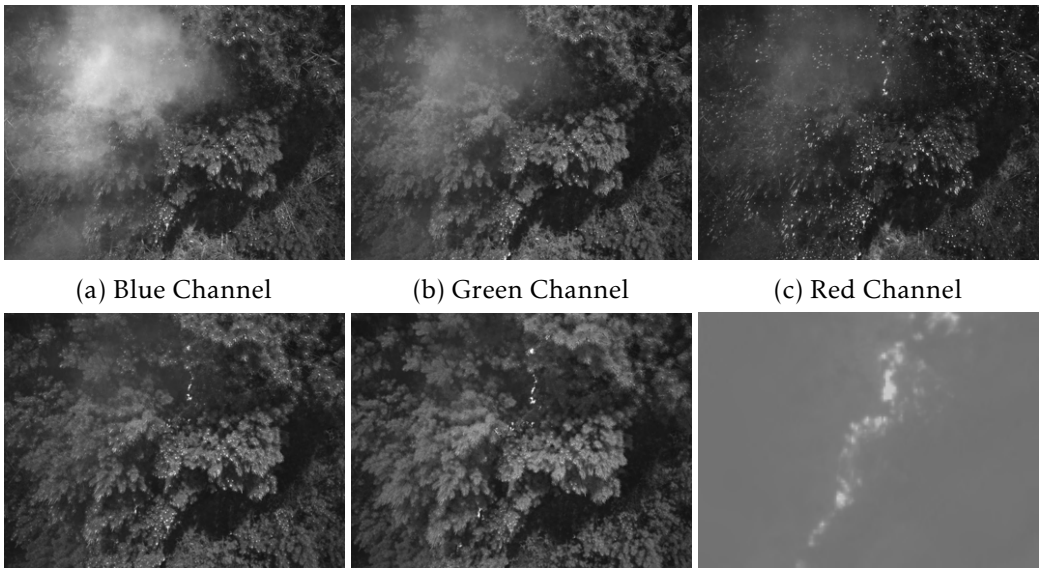
As it was mentioned earlier, in section 2.4.2, in order to train a network capable of detecting objects with an accuracy and precision suited for a real world situation, a robust data set is crucial. This data set needs to be constituted by a large amount of images containing a wide variety of examples, aiming to represent the range of different cases that are possible to be encountered on the field. A data set composed by aerial images of several different fires captured by a multi band sensor with a channel for each different wavelength, from blue to TIR, would be the ideal. However, this type of specific data is usually used for research purposes and it is very difficult to be found online. Because of this, the data set used in this project was exclusively composed of data collected from prescribed fires, which were made possible thanks to the FoCoR PCIF/MPG/0086/2017 project, funded by Fundação para a Ciência e Tecnologia (FCT).

Seven different groups of images were captured, each one representing a different fire. Some of these sets were captured in the same day with fires in similar environments, while others captured fire occurrences with totally different surroundings. The images with fire instances were captured with altitudes varying between 10 and 120 meters. A RICS UAV equipped with a MicaSense Altum Multispectral camera was used to collect the multi spectral images presented in all data sets mentioned in this dissertation. Each captured image is divided in six channels, providing 6 different tiff files, with each one representing the data in the Blue wavelength, Green, Red, Red Edge, Near-IR and Thermal. All the image files are 1 channel 16 bit tiff files with 2064x1544 pixels except for the Thermal data, with only 160x120 pixels[65].

It is worth noting that all 6 images are not completely aligned and it is very difficult to, for example, obtain the RGB image through the direct merging of the first three channels. An example of two different channels from the same image are overlapped and displayed in 4.3a to show a possible difference in the channels data. In figure 4.3 it is also presented the same image's RGB channels directly overlapped and the actual RGB image side by side.

The thermal channel has also a different and wider field of view of 57°x44° compared to the other channels of 48°x37°. This aspect is exemplified in figure 4.4. It can also be observed in figure 4.2, where the most intense burning areas visible mostly in the Red Edge and Near-IR bands are not as close to the centre of the image as in the Thermal band. The Thermal band image covers a vaster amount of area indicating its wider field of view.

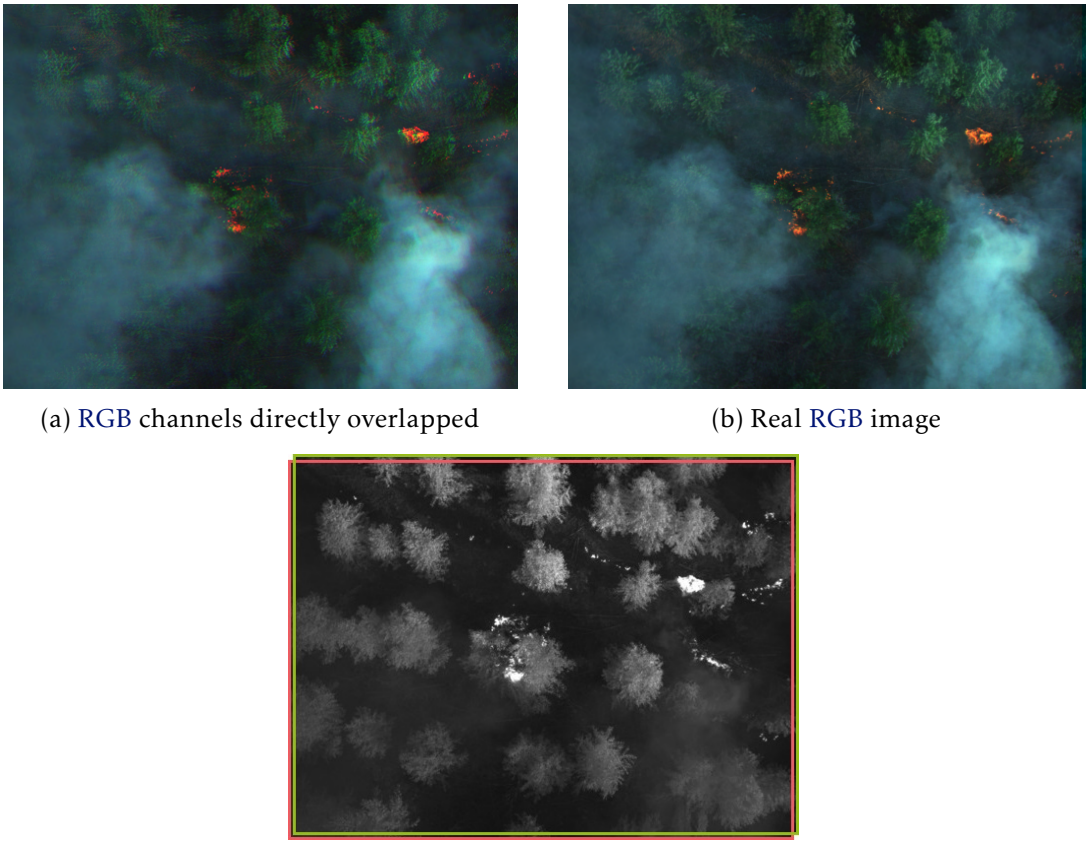
The collected images were divided in different sets, each set corresponding to one



(a) Blue Channel (b) Green Channel (c) Red Channel

(d) Red Edge Channel (e) Near Infra Red Channel (f) Thermal Channel

Figure 4.2: Image example on the six different channels.



(a) RGB channels directly overlapped (b) Real RGB image

(c) Red Edge (Red frame on the bottom) and Near-IR (Green frame on top) deviation

Figure 4.3: Examples of an image's different bands misalignment.

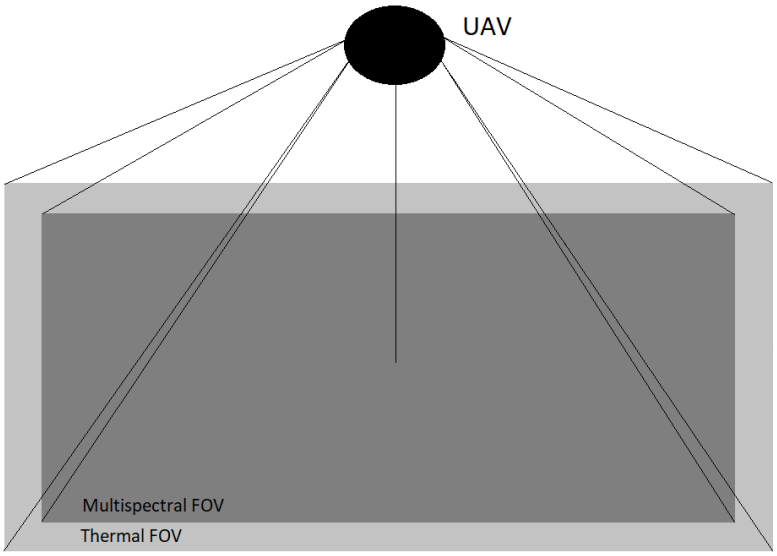


Figure 4.4: Example of Multispectral (darker grey) and Thermal (lighter grey) different Fields of View.

channel. The three last channel sets, Red Edge, Near-IR and Thermal, are the channels that highlight fire occurrences the most. The sixth channel exposes all high temperature areas and the fourth and fifth channel, compared with RGB, are the least sensitive to smoke and where flames are the most accentuated. These last three channels were selected to individually train the network and their different results are compared in the results Chapter 5. To acquire equivalent results worthy of comparison, the training for all three channels needs to be performed in equal conditions. Each channel set was divided in two sets, training set with 834 images and validation set with 213. All three different channel sets had the same group of images in their training and validation sets to ensure equal circumstances. After this preparation step, all images in these different sets were carefully labelled.

4.2.2 Image Annotation

When training an image classification network, every data set needs to contain the necessary information with the pixelwise area where objects are located within each image. This information can be represented in different formats, depending on the framework, and to obtain it, an annotation process needs to be performed. With the data sets organised by their channels, every image needs to be labelled according to where fire regions are located. However, the multispectral images are all grayscale which can make it difficult to the human eye to detect particular fire occurrences.

Before starting labelling images, an orthomosaic image was obtained to help identify fire regions. This RGB image contains all the captured images correctly scaled and aligned, and helped noticing some smaller flames for example, that otherwise could be missed. These orthomosaics were obtained with the help of Agisoft Metashape[66], a

software application specialised in digital imaging processing. An example of one of the seven groups of images orthomosaic is displayed below in figure 4.5. Even though these orthomosaics may exhibit image distortion in some smaller areas, this is still considered an advantageous tool by displaying the overall coloured data set, avoiding converting each image to RGB individually.

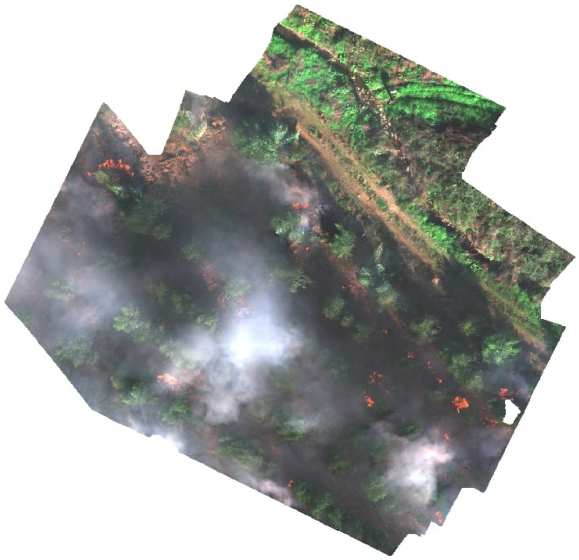


Figure 4.5: Orthomosaic example.

The images annotations on Mask R-CNN are required to be represented in a single JSON file per set, with a JSON object for each image in the set. Having this in consideration, a tool with the option to extract the annotations as a JSON file, VGG Image Annotator (VIA) [67], was chosen to help in the image annotation process for this framework. This is an open source manual annotation software for image, audio and video and runs as an offline application in a web browser. This tool was opened via Microsoft Edge, which is, by the time of the writing of this dissertation, the only browser, besides Safari, capable of displaying tiff files.

This process started with importing all the images from the data set and choosing the image files' directory as default to simplify the loading step after every break. VIA has the option to download a JSON save file, so that the user is not forced to label all the images in the set in one go. Next, an attribute "class"was created with one element called fire, the only relevant class in this project. When defining a region on a chosen image, that region can be manually classified as one or multiple elements of the class attribute. Starting by the first image, a polygon shaped region was delineated around every fire occurrence and defined as the only class element, "fire", each one with an index number automatically associated when created. This process was then repeated for all the images in the data set and, when finished, the JSON file containing all the information regarding that data set annotation, was exported. Each image and its annotations are detailed in the JSON file with the structure presented below:

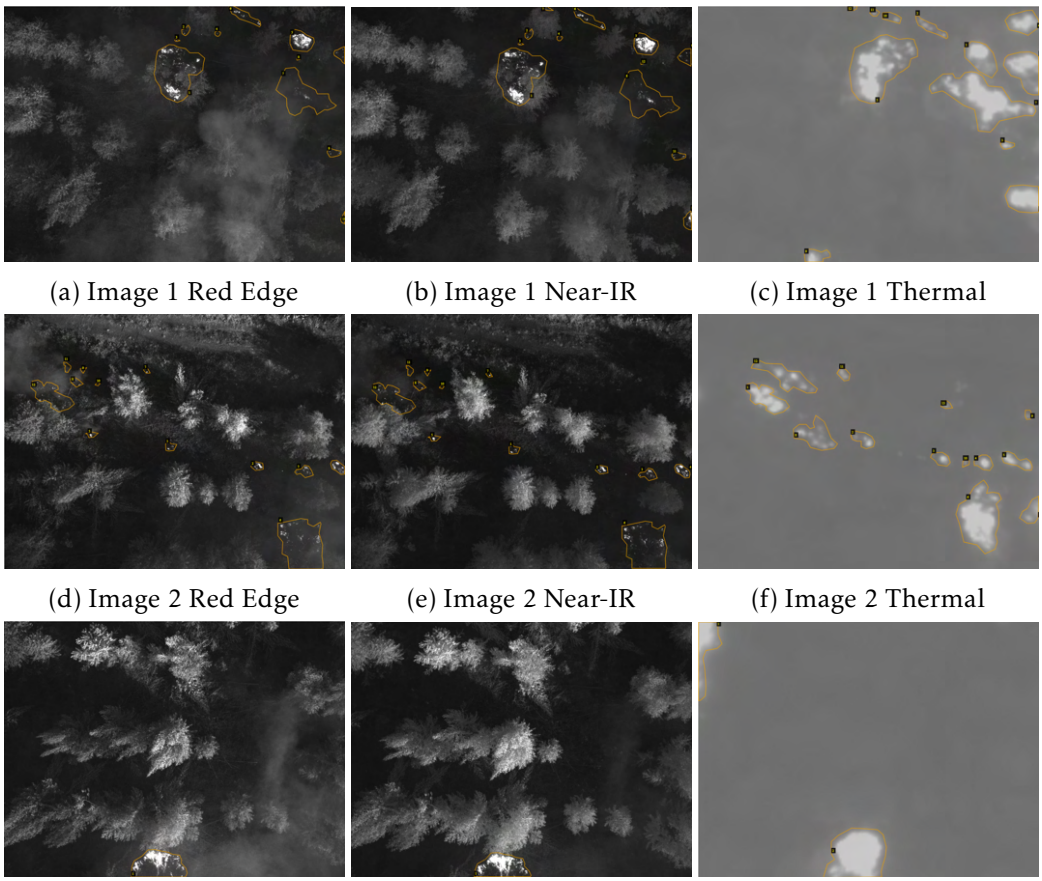
- **filename:** Contains the name of the image file.
- **size:** Value specifying the size of the image file in bytes.
- **regions:** Lists all regions drawn in the image file. Each region is defined by a specific set of attributes:
  - **shape_attributes:** Includes the name of the shape that determines what the next attributes are. For this data set the only shape used was the polygon, which is described by the coordinates of all its edges, in the x axis, all_points_x and y axis, all_points_y. If, for example, the rectangular shape was used, it would be described by the coordinates of the top left edge, width and height values.
  - **region attributes:** Holds the name of the class represented inside the region.
- **file_attributes:** Optional feature included in VIA that allows to add extra information to the file with no direct link to the regions. This option was not used in this annotation process.

As mentioned earlier in 4.2.1, the MicaSense sensor collected data is not, by default, correctly aligned (figure 4.3), and the Thermal data does not have the same size as the other spectral bands' images nor the same Field of view (figure 4.4). This implies that when collected an image, this image in the fourth channel for example, does not correspond entirely to the same image in the fifth channel, and so, the annotations generated for one channel can not be reused for the other channels. The three different data sets were individually annotated. Some examples of these annotated images are presented in figure 4.6.

4.2.3 Parameters

The annotation files as well as the data sets were imported into Google CoLab where the training process was performed. CoLab is a hosted Jupyter notebook that allows the user to execute python code through the browser[68]. It provides free access to GPU computing, making it ideal for machine learning. It can also be connected directly to Google Drive, where all the data sets were located. CoLab GPU Virtual Machine has a maximum run time of 12 hours, meaning that the network can never be trained more than 12 hours straight. It can however, re-start from the last completed epoch without losing previous work.

When using CoLab, the data sets as well as the Mask R-CNN itself need to be firstly downloaded to the environment, remaining there for the next 12 hours. These circumstances and the fact that each band training would present different feedback influenced the author to train one band before the rest. Only the data regarding the Near-IR channel was imported first, which was used to tune all the important parameters necessary to



(a) Image 1 Red Edge (b) Image 1 Near-IR (c) Image 1 Thermal

(d) Image 2 Red Edge (e) Image 2 Near-IR (f) Image 2 Thermal

(g) Image 3 Red Edge (h) Image 3 Near-IR (i) Image 3 Thermal

Figure 4.6: Annotation examples.

train the model in the most efficient way and with the best possible results. When this process was finished and the network was presenting its best results, the other two bands started their training processes.

A python file, fire.py, was created to train the network, based on other training files already on the framework. The majority of the frameworks tuned parameters were overwritten in this file, avoiding replacing directly their default values. The tuned parameters as well as the used values to train the network are listed below. Most of these parameters were altered multiple times but the values displayed were the optimal ones that led to the best achieved results. These values are all displayed also in table 4.1 below.

- **num_classes:** Number of classes the network will be capable to detect. This number is always set to the number of trained classes, in this case 1 for the fire class, plus 1, considering the background class.

- **detection_min_confidence:** Minimum percentage value of confidence for which the network considers as a detection. This value was set to 0.9, skipping all detections with less than 90% confidence.

- **images_per_gpu:** Number of images loaded at the same time by the network. To optimise and reduce training time, this parameter was set to 2, which proved to be the most appropriate value for the 12GB GPU from Colab.
- **steps_per_epoch:** Number of training steps each epoch performs. It was set to 200 steps.
- **validation_steps:** Number of validation steps each epoch performs. It was set to 100 steps.
- **epochs:** Number of epochs to be performed during training, which oscillated mostly between 100 and 130. This parameter along with steps_per_epoch and validation_steps varied the most, with these presented values proving to be the best compromise between training time and results.
- **learning_rate:** Value representing each network update significance during training. This parameter was firstly reduced from 0.01 to 0.001, lowering the impact of each update during the training process and vastly reducing the probability of skipping the optimal point.
- **layers:** Parameter indicating which architectures to train, between Head and Backbone + Head. This model will be responsible for detecting one class in grayscale images, which is completely different from the pre-trained COCO network. Since the Backbone architecture is the one used for feature extraction over images, both architectures were selected by setting layers='all'.
- **backbone:** Parameter specifying the backbone network architecture to be used between Resnet 50 and Resnet 101. It was set to use Resnet 101 to obtain the lowest error rates possible[69].
- **image_min_dim** and **image_max_dim:** Re-scaling parameters where image_min_dim represents the size in pixels of the smaller re-sized image and image_max_dim the maximum size that the longer side can reach. After re-sizing, the image is padded with zeros to establish a square shape and ensure multiple images can be loaded at the same time. The image_min_dim parameter was set to 800 and the image_max_dim to 1024, ensuring that the training was not overly long without losing too much image information.

The Mask R-CNN framework is built in a way to minimise training time by requesting a file with pre-trained weights such as the ones from COCO data set. When loading the weights, the first layer, conv1 is excluded, initialising this layer weights randomly and preventing conflict between the 1 channel input images and the weights generated with 3 channel images from COCO.

In order to obtain a wider variety of examples to train the network, data Augmentation was performed with imgaug[70], a library for image augmentation. Data augmentation,

used for training in machine learning algorithms, is a wide variety of techniques applied to already existing data in order to expand it. The existing data is copied and modified to join the data set with no duplicates. For image data, some of these techniques are Gaussian Noise, uniform blur and translations but for the acquired data sets, only procedures that did not involve altering pixels intensity were used. With imgaug, labelled RoIs are also copied and suffer the intended alterations, meaning that this process can be applied after annotating the original data sets. Before introduced in the network, each image and its labelled boxes have approximately an 80% chance of being copied and suffering a transformation such as horizontal and vertical translation, rotation and scaling.

Table 4.1: Parameters values used to train the network.

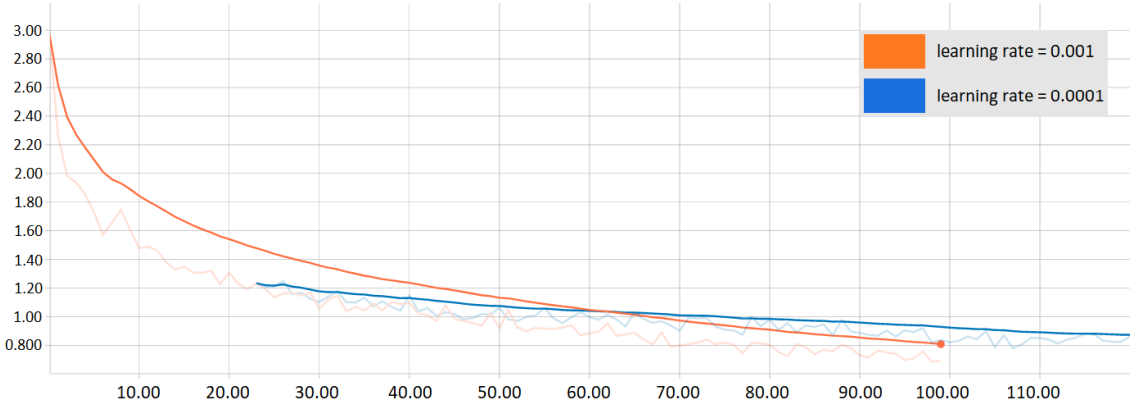
Parameter	Value
num_classes	2
detection_min_confidence	0.9
images_per_gpu	2
steps_per_epoch	200
validation_steps	100
epochs	100-130
learning_rate	0.001/0.0001
layers	Head and Backbone
backbone	Resnet 101
image_min_dim	800
image_max_dim	1024

4.2.4 Training

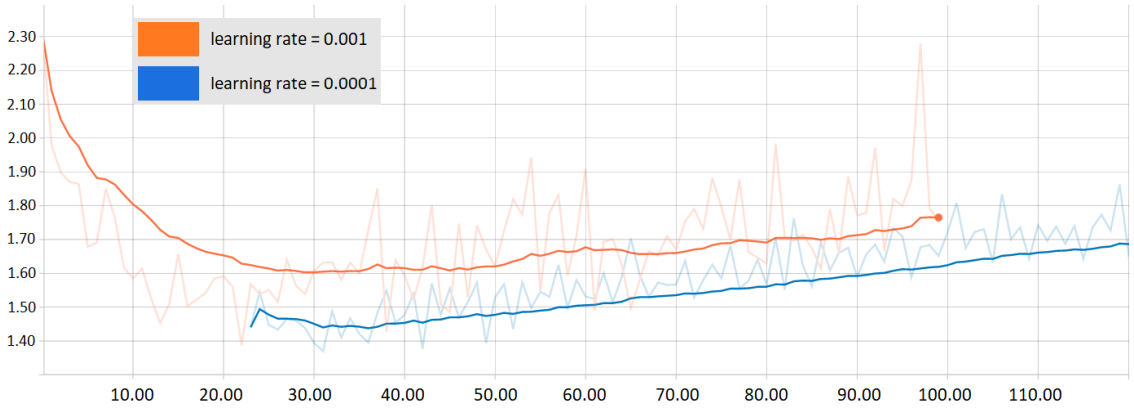
To start training, the function train inside fire.py was called with the path to the training and validation data sets. This function ran the train method from model.py with the following input parameters: the data sets, the random image augmentation effect, the learning rate value, the number of epochs and the layers to train.

The Mask R-CNN model covers the entirety of the training set during each epoch, and when finished, runs trough the validation steps. While the model is training, the Mask R-CNN network also determines and returns the training loss values after every step and the validation loss value after each epoch, making it accessible to follow its progress.

An example of a tensorboard graph displaying the training process evolution is shown in figure 4.7. Both represented loss values are the sum of rpn_class_loss, rpn_bbox_loss, mrcnn_class_loss, mrcnn_bbox_loss and mrcnn_mask_loss. Each of these loss metrics are also, in turn, the sum of all the loss values calculated individually for every RoI. Although a continual decrease is observed in the training loss values, around the 26th epoch, the validation loss values stop decreasing and start increasing again around epoch 47. This overfitting manifestation informed that the best HDF5 weights file to be used in the model is contained in that zone where the validation loss value is the lowest and the most



(a) Total training loss



(b) Total validation loss

Figure 4.7: Near Infra Red channel training behaviour.

consistent. However, there is still the possibility that the network has skipped the optimal point. To minimise this probability, the network was re-trained with a different learning rate value of 0.0001 starting from epoch 23, which showed the smallest validation loss value. This re-training process evolution can be observed by the blue line.

With the most satisfactory loss values overall, the mask_rcnn_fire_32.h5 file from the lowest learning rate train was downloaded and later introduced in the detection phase. With the best and most efficient way to train the network identified, this training method was performed after for the other two channels, Red Edge and Thermal, where the results were also evaluated with tensorboard graphs and the best weights files downloaded. This process is further detailed in 5.2.

4.3 Detection

To perform the inference process with one of the files obtained in the previous phase, a python file, inference.py was created. This receives as input the path to the HDF5 file, the path to the input image, and the path to where the output image with the overlapped mask is going to be saved. Initially in this file, like in fire.py, the number of images per

GPU is overwritten so that depending on the GPU memory of the machine running the detection model, the inference time can be greatly reduced. The number and names of the classes to be detected are also defined here, which can be useful in future applications where there may be instances of more classes other than fire to be detected. With this concluded, the Mask R-CNN model is then initialised in inference mode and loads the input HDF5 file as the network's weights trough the load_weights function.

This inference process was organised to be performed on specific multi spectral tif images. As such, the single channel tif file received as input by the model cannot be directly fed to the network and its output displayed, since it would only result in consistently detecting zero instances. When reading the image, a copy of the original image is also created and instantly converted from 16bit format to 8 with img_as_ubyte function from skimage library [71]. This copy is then converted in a RGB file by placing the same 8 bit value from each pixel in all of the 3 channels with the help of cv2 libraries function, cvtColor. The unaltered original image is run through the Mask R-CNN model's detection function which returns the results in 4 arrays: rois, class_ids, scores and masks. After finishing this, the display_instances function in visualize.py is called to return a matplotlib plot with the input image and the coloured masks contained in the arrays overlapped. This display_instances function in visualize.py was modified to receive the path where the image output is saved and, before showing the output in a plot tab, saves it as a PNG file. When called, the 4 arrays are sent as arguments as well as the copied altered image to be displayed, the previously defined class names to be displayed next to the detected instances and the path where the output is to be saved.

4.3.1 Thermal Band Inference

As discussed and displayed earlier, the thermal band data is unique from the other five bands and despite being suitable for fire detection applications, it is not perceptible to the human eye. When performing the inference phase with the thermal band, the mask cannot be overlapped in the thermal image, or the user reading the output will not be able to understand where the fire is actually located.

To tackle this issue, the mask is overlapped in an image that the user can interpret. The model will still detect the fire regions with the thermal data, receiving the thermal image in the detection section, but sends the image in a different band to the Image Conversion section. The Near-IR band was selected to be the overlapped image, since it is the band that is the least sensitive to smoke (explored in section 5.3), allowing for a clearer view of the captured field.

The two different band images have different sizes and fields of view and the masks would be incorrectly located if these were just overlapped like in the more standard process. These images require adjustments in the Image Conversion section in order to align them accordingly. However, the proper camera calibration process is a complex operation that requires a lot of images and needs yet to be implemented. For this model,

a different and specific alignment process was implemented. It accomplishes its goal for the images used in this dissertation, but cannot be considered an ideal method. As it is mentioned later in 6.2, it is necessary to implement a solid calibration method that allows aligning the captured images beforehand.

To start performing this alignment process, the thermal image is resized to the same proportions as the Near-IR band and the borders cropped accordingly, due to the thermal sensor wider field of view. There is also a visible displacement between the thermal sensor and the Near-IR sensor and therefore, the thermal image needs to undergo a down translation and a left translation. After this process is completed, the thermal image is correctly aligned with the Near-IR image.

The detection process is later performed in the obtained image, and the masks are obtained. Having the thermal masks correctly aligned with the Near-IR image, the overlapping process is executed and a Near-IR image with the coloured masks around the high temperature areas is returned.

4.3.2 Rekindle Inference

As explained before, it would be extremely useful for a fire detector to be capable of detecting possible rekindling occurrences, which would in turn prevent future fire outbreaks. One first approach to accomplish this should be detecting all areas that are at extreme temperatures but do not show signs of being currently burning. A suggestion to achieve this goal was implemented.

Considering that the Red Edge or Near-IR based models detect all visible (even through smoke) burning instances, and that the Thermal based model detects all high temperature areas, subtracting the visible areas from the high temperature areas would result in the desired areas. To produce this result, a new python file was created where both inference and saving processes are performed like in the inference.py file. In this case, 2 images are received, Thermal and the other chosen band, as well as 2 HDF5 files, one corresponding to the thermal model and other to the other band. The inference process is performed on both images, and two 3D mask arrays are computed. The same method used for the thermal band inference is applied here as well to align both thermal and the other band images and masks.

Two empty 2D boolean arrays with the size of the received image are created with only False values. To one of the arrays is added the mask values of the Thermal detected instances, turning every value where the thermal masks are located into True. This process is then executed for the other 2D array and the visible fire masks. This results in two image size masks with all the instances for each one of the images detected instead of individual bounding box sized masks for each instance.

A logical XOR is applied to these two masks, resulting in a mask with all the areas where only one of the images detected an instance, but not both. A logical AND is applied right after, between the resulted mask and the thermal mask to ensure that the presented

instances belong only to the thermal detection. When this is over, the resulting mask is overlapped with the visible chosen band image to present a perceptible map of where the possible rekindling areas are located. This Rekindle Inference process (with a Near-IR image) as well as its result is presented in the figure 4.8 below.

With the Covid-19 global pandemic, there were an extensive amount of resources that were limited during the writing of this dissertation. An example is the extra amount of images necessary for further training and even for calibration, that were never collected. Many implemented processes have also become more time consuming because of the lack of resources resulting from confinement. Due to these constraining factors, it was not possible to implement the proposed solution where the high temperature areas went through a thorough selection process. Ideally, the obtained rekindle areas would not include, for example, locations too close to burning areas. Unfortunately this method was not achieved in this dissertation and it is suggested in the future work section 6.2.

It is important to mention that the post Mask R-CNN inference images are only included in the schematic to help visualise the rekindle detection process. Despite presenting the obtained masks for each band, these do not correspond to the actual detections' outputs, which are only composed of arrays, with no visual data.

The obtained result displayed in the schematic is a good example of the implemented rekindle detection. The red masks correspond to all high temperature areas that were not detected in the Near-IR classification. In this case, if the proposed solution was fully implemented, the areas that could be considered as rekindle instances would be the two major and more central ones, but not the area around the large burning instance at the bottom of the image.

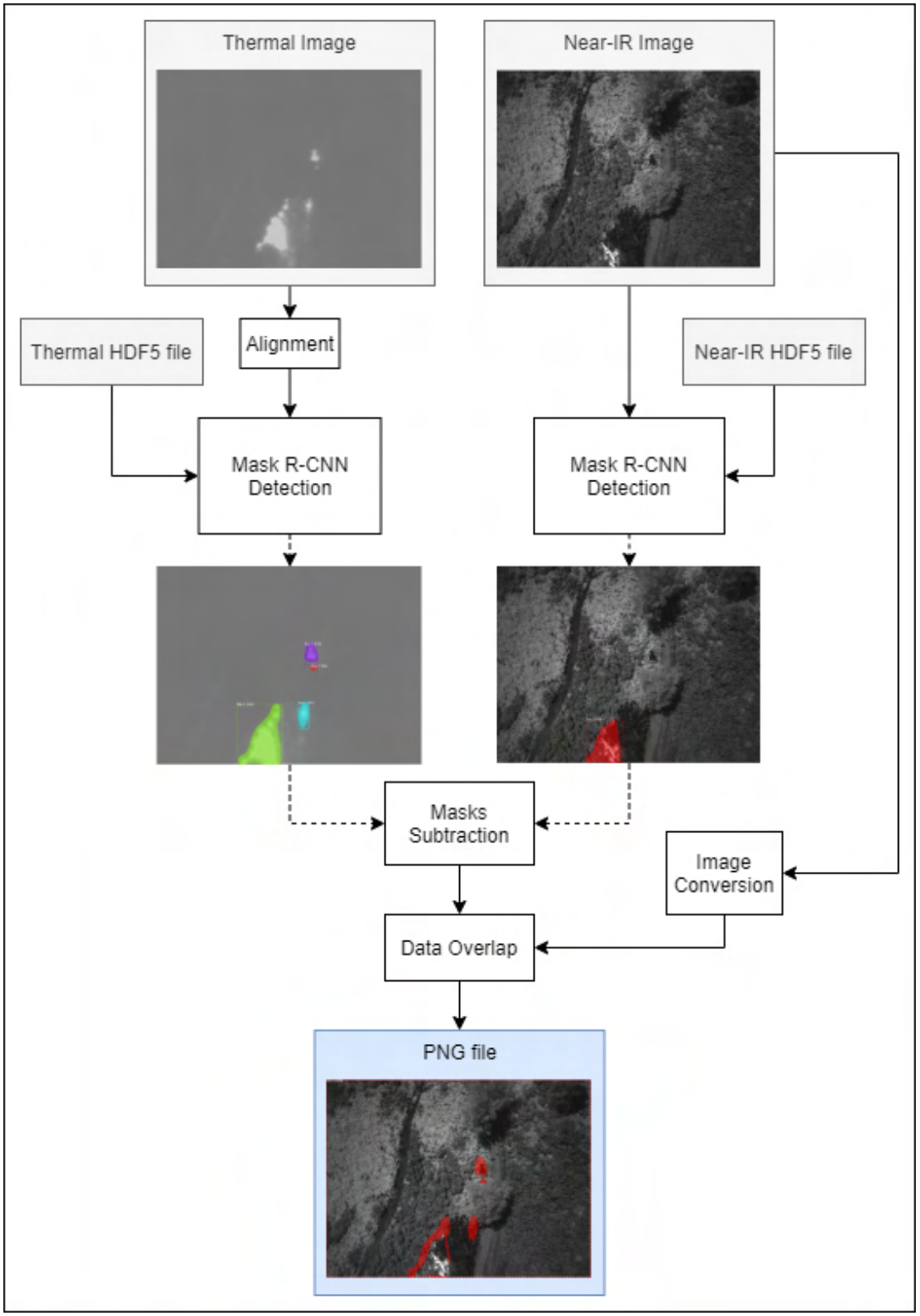


Figure 4.8: Rekindle Detection diagram example.

Chapter 5

Experimental Results

This chapter presents, in detail, the implemented procedures responsible for acquiring data that, in turn, allows an in depth evaluation of the implemented model performance. It starts, in the first section, by presenting the results that support the Mask-RCNN framework choice for this model. The second section of this chapter is mainly focused on the training performed with the different data sets and the third presents the model's inference outcome. Both training and inference results are analysed, compared and discussed to interpret their overall quality and impact on the model's final product.

5.1 YOLO vs Mask R-CNN Benchmark

As mentioned before in chapter 3, to implement a fire detection model, an object detection framework is necessary to be included in the model. However, there are multiple frameworks designed with similar objectives, some of these are briefly explained in 2.4.5. This section explains the important characteristics that make a detection framework relevant to this model and why the Mask R-CNN was the selected between the possible candidates.

The most appropriated framework should be one with both satisfying accuracy and speed, which ideally could be identified through framework comparison studies. Unfortunately, most studies found present different groups of frameworks and are usually performed under different environments with distinct test sets and hardware. A series of studies had to be considered and compared, to try and infer the most adequate frameworks. The table below is an example, collected from [56], and displays the average precision from some of the most recent and advanced object detection frameworks such as YOLOv3 and Faster R-CNN.

Although the RetinaNet presents the best results in terms of precision, it also is the

	backbone	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
Two-stage methods							
Faster R-CNN+++	ResNet-101-C4	34.9	55.7	37.4	15.6	38.7	50.9
Faster R-CNN w FPN	ResNet-101-FPN	36.2	59.1	39.0	18.2	39.0	48.2
Faster R-CNN by G-RMI	Inception-ResNet-v2	34.7	55.5	36.7	13.5	38.1	52.0
Faster R-CNN w TDM	Inception-ResNet-v2-TDM	36.8	57.7	39.2	16.2	39.8	**52.1**
One-stage methods							
YOLOv2	DarkNet-19	21.6	44.0	19.2	5.0	22.4	35.5
SSD513	ResNet-101-SSD	31.2	50.4	33.3	10.2	34.5	49.8
DSSD513	ResNet-101-DSSD	33.2	53.3	35.2	13.0	35.4	51.1
RetinaNet	ResNet-101-FPN	39.1	59.1	42.3	21.8	42.7	50.2
RetinaNet	ResNeXt-101-FPN	**40.8**	**61.1**	**44.1**	**24.1**	**44.2**	51.2
YOLOv3 608 × 608	Darknet-53	33.0	57.9	34.4	18.3	35.4	41.9

Figure 5.1: Different frameworks precision[56].

slowest, achieving speeds not compatible with real time detection. The SSD variants, despite presenting similar precision results to the ones from YOLOv3, are also presented as 3 times slower in [56]. Faster R-CNN and YOLOv3 present the best Average Precision (AP) values from the frameworks presented in 2.4.5. Mask R-CNN on the other hand, has practically the same architecture as Faster R-CNN with FPN and shows results with the same speed and precision, adding the mask to the output image. Object segmentation is extremely important for the purpose of this dissertation since it may specify where the burning area is localised and not only the square shape area where it is occurring. This is critical mostly when the captured images present massive fire areas, causing the big generated bounding box around them to be irrelevant. However, YOLOv3[72] presents the highest speed rates, with almost double as Mask R-CNN [73], which, together with its high accuracy, may be a decisive factor.

Since YOLOv3 and Mask R-CNN are the two frameworks presented in 2.4.5 with the best results overall for the COCO data set, these were the frameworks first considered to integrate the implemented model. Both models networks were trained to this dissertation specific problem to conclude which one is the more suited. Both used the same data set (explored in 4.2.1) and were trained with Google CoLab notebooks (4.2.3).

These two frameworks work with different types of shapes, where Mask R-CNN recognises all kinds of polygons to cover the detected objects and YOLO only rectangular boxes where the objects are inserted. Because of this, these frameworks require different formats of annotations for the training and validation sets and thus, the process of image annotation was divided in two and the same images had to be labelled twice.

The darknet architecture from YOLOv3 currently does not support TIF files and so, the full data set had to be converted to either PNG or JPG beforehand. On this model, the annotation format is essentially a TXT file for each one of the images contained in the data sets, with each file structured with one integer and four float values per object. The integer indicates the class that the object belongs to (text file holds all classes to be trained) and the float values represent the area where the object is located within the image.

To simplify the task of obtaining the correct float values for each instance of an object, labelIMG, a graphical image annotation tool, was used [74]. This tool, written in python, has the option to graphically insert a box around the areas where each object is inserted, assigning it a class previously defined. It has also the option later to save the annotation file in the YOLO format, which allowed this complete process to be finished in a reasonable time period. The annotation process for the Mask R-CNN model was explained in detail in 4.2.2. Since the annotation process was necessary to be performed twice, only the Near-IR band data was labelled and used for training. If the results proved to be insufficient to draw a conclusion, the other bands would also be tested and compared.

After trained, both networks performed the inference process on a test set which is described in more detail in section 5.3. The YOLO framework, like Mask R-CNN, has the tools necessary to calculate the Mean Average Precision (mAP) (section 5.3) of a trained model. Both networks mAP were computed and are displayed below.

Table 5.1: Mean Average Precision with IoU=0.5 for both Frameworks.

Framework	mAP
Mask R-CNN	43.2
YOLOv3	37.0

By comparing both values, it can be concluded that the YOLO framework, even with a lower level of complexity and no segmentation process, presents inferior detection results. The inference times presented by Mask R-CNN (later in section 5.3) are satisfactory for a real time multispectral fire detection model, which does not require extremely fast responses. As such, the YOLO network does not present an extreme advantage with its faster detection and does not demonstrate an ideal level of precision. These circumstances indicate that this framework is not as well suited to integrate the proposed model, and cannot overcome the value that the Mask R-CNN instance segmentation adds to the system. As such, the Mask R-CNN was the framework chosen to integrate the model proposed in this dissertation.

5.2 Training Evaluation

As mentioned before in 4.2.4, during the training process, all loss values were computed and a plot for each one was delineated. All three channel networks were trained multiple times with the parameter's values listed in section 4.2.3.

The graphs displayed in the next subsections all represent the behaviours relative to the training procedures in which the best results were obtained and not to the ones with the most typical behaviours. These are all concerning the validation losses, as the training loss graphs all present similar behaviours to the one displayed in the previous chapter, in 2.4.2. The smoothing effect is present in every tensorboard graph to emphasise its

tendency and overall progression. Nevertheless, the real loss values can also be observed slightly faded in the background.

The learning_rate was decreased for every training to 0.0001 and trained afterwards for this value multiple times as well. This decrease was mostly practised during and after the training's overfitting behaviour. None of the training processes displayed occurred for longer than 120 epochs.

5.2.1 Near-IR Training

In figure 5.2 it is shown the total validation loss to display the general training behaviour, as well as the Masks' and Bounding Boxes' validation losses. Despite the existence of more loss metrics, these two were the ones chosen to be displayed, since the Bounding Boxes around the RoIs and the masks overlapping the objects are the main focus of the presented model.

By looking at any of the graphs, it can be recognised that, as mentioned in 2.4.2, the overfitting behaviour occurred around the epoch 26 where the validation loss stopped decreasing, unlike the training loss. Both Mask and Bounding Boxes present similar courses where the decrease of the learning rate proved to be beneficial. Even though it did not prevent the overfitting behaviour completely, it managed to achieve, in average, smaller loss values.

With each epoch taking between 250 and 320 seconds, every Near-IR network training took in average 9 and half hours to be complete.

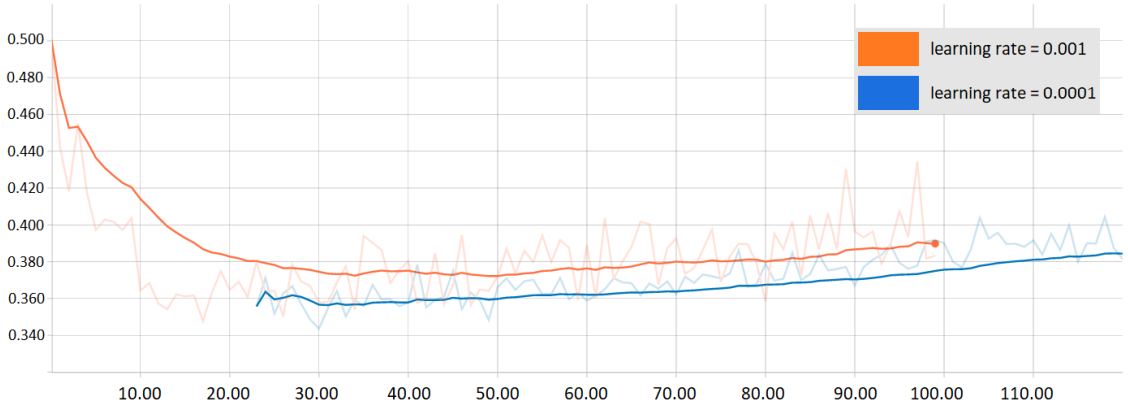
5.2.2 Red Edge Training

The Red Edge channel training progress is displayed in figure 5.3 and resembles roughly the Near-IR one. In this case, the training started overfitting around the 45th epoch which can be concluded by looking at the mask loss behaviour. The bounding box loss takes a longer number of epochs to stop decreasing, starting around epoch 82. However, by that time, the mask loss value is already higher than desired.

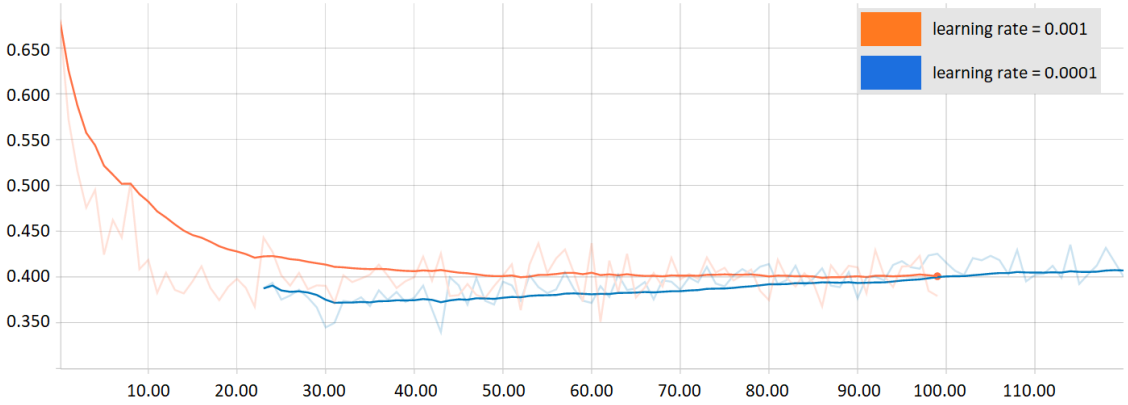
Like in the previous channel, lowering the learning rate carried an improvement, with the best epoch overall being the number 62. However, this lowest loss value is considerably higher than the ones obtained with the Near-IR channel. This will result in a less reliable model with a weaker accuracy and precision.

There are multiple reasons that may explain this difference between these results since CNN's logic can be rather complex to comprehend. One aspect that differentiates the two channels and may be key to justify this discrepancy is that in the Near-IR channel the flames have a greater pixel intensity and stand out more from the different surrounding environments. In the Red Edge band, flames can blend in and be eventually mistaken by reflections, having a great impact on the models detection confidence.

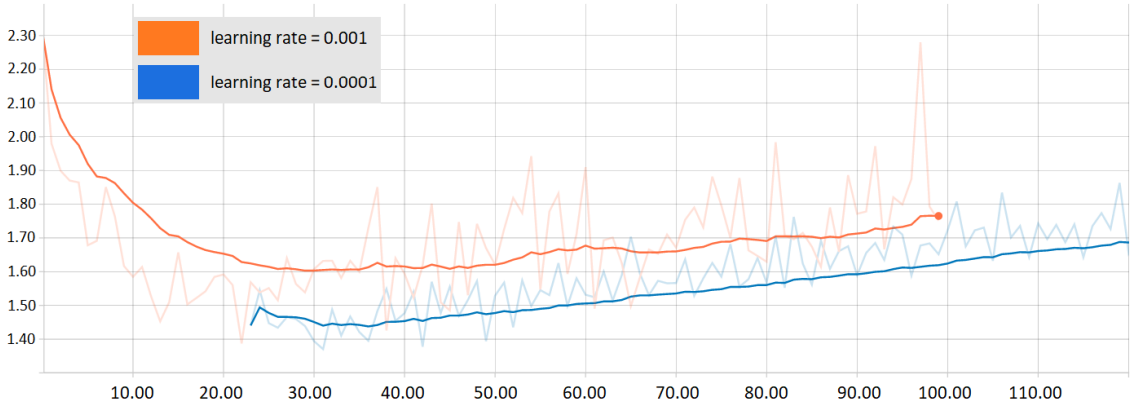
This training process is performed in an average of 9 hours, making it the fastest from the three different training processes.



(a) Masks validation loss



(b) Bounding Boxes validation loss

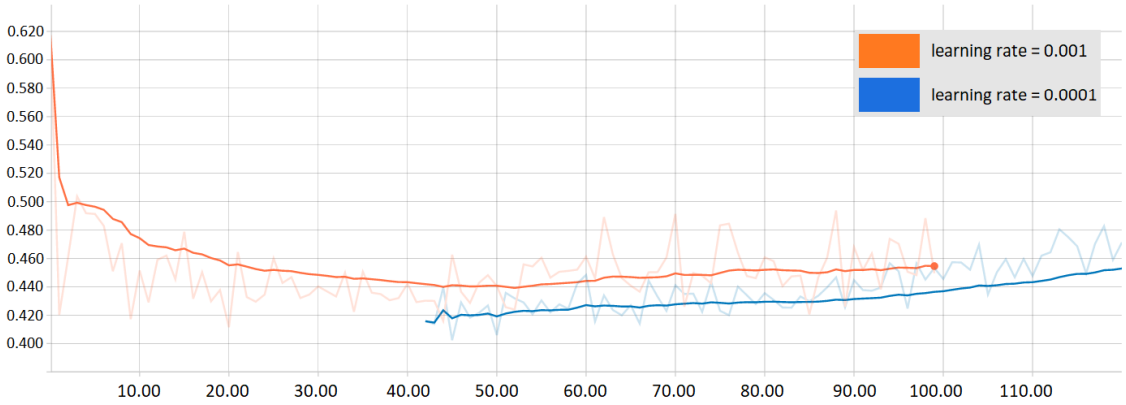


(c) Total validation loss

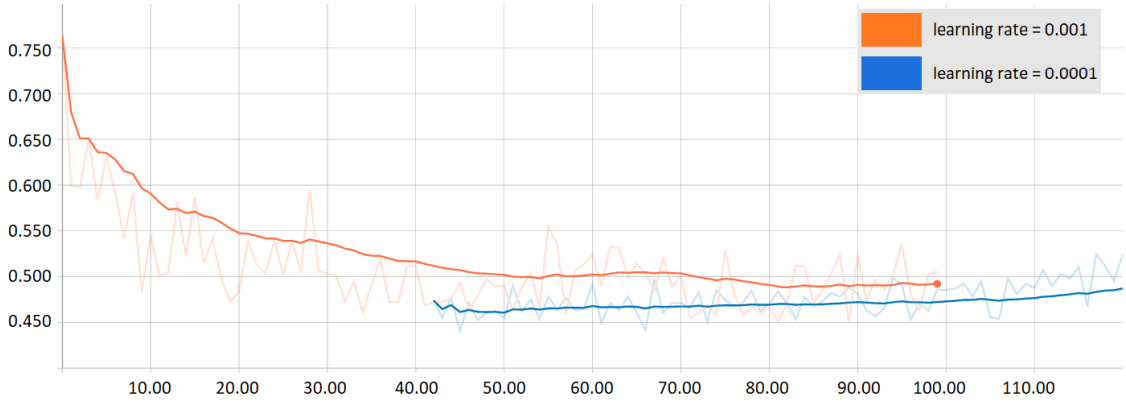
Figure 5.2: Near-IR channel training behaviour.

5.2.3 Thermal Training

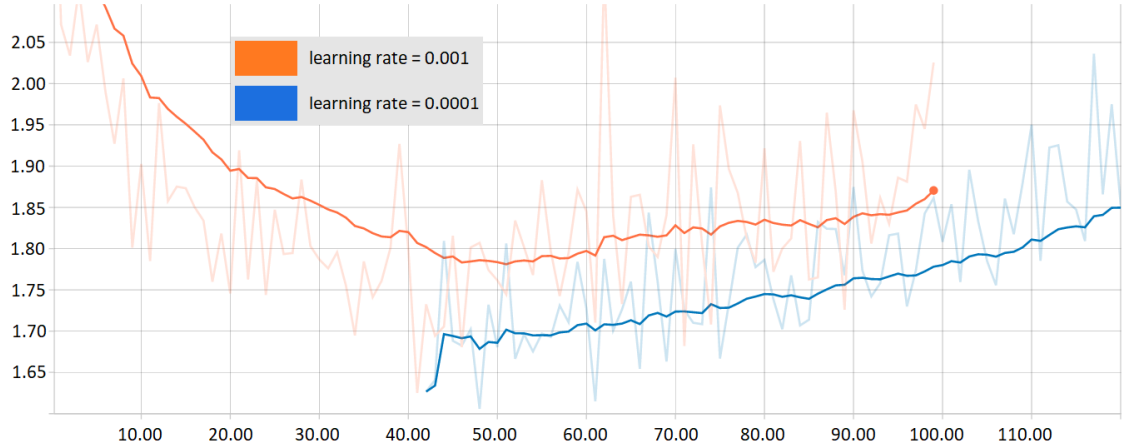
When observing the graphs in figure 5.4, the most noticeable aspect is that the loss values sum is fairly inferior to the ones presented before. Both mask and bounding box loss graphs have starting values lower than the ones presented as the lowest in the Near-IR channel. It is also worth noting that the training loss graph, despite showing the same general direction and movement as the other bands, also starts at a lower value, below



(a) Masks validation loss



(b) Bounding Boxes validation loss

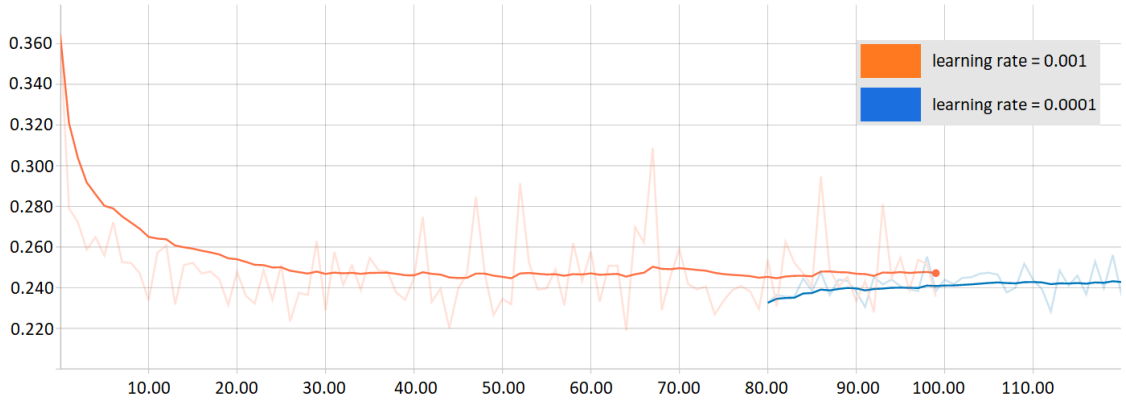


(c) Total validation loss

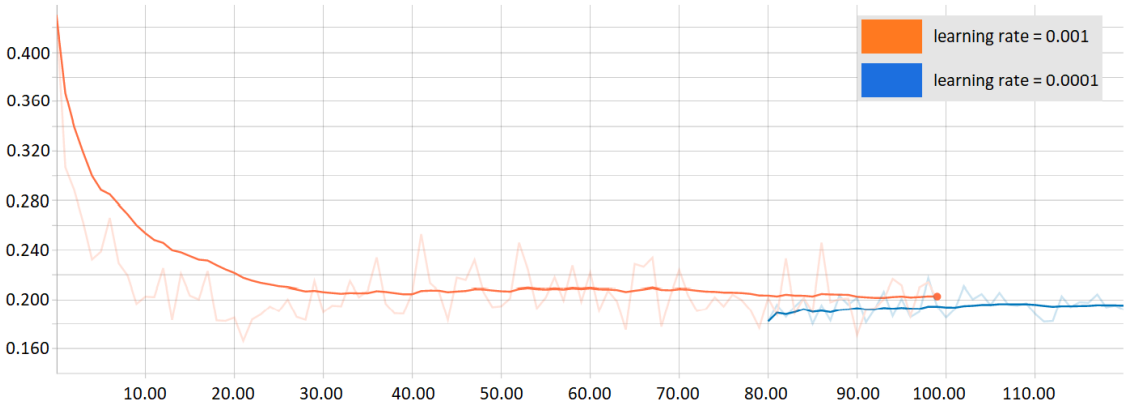
Figure 5.3: Red Edge channel training behaviour.

1.5.

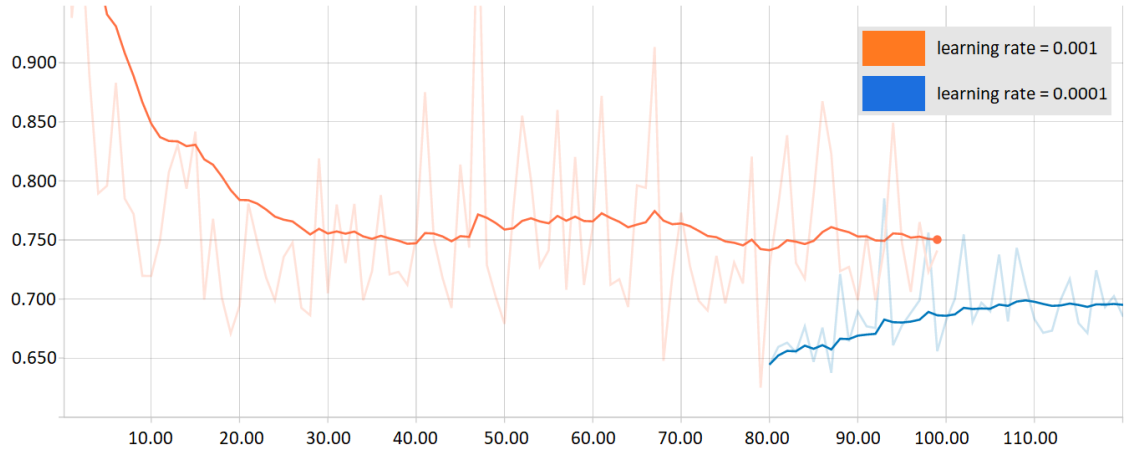
From all three different bands, the Thermal band was the one that presented the most atypical training behaviour regarding validation losses. This may be due to the fact that the majority of parameter tuning was performed for the fifth channel and the Thermal band presents visible characteristics that differentiates it from the other channel's data.



(a) Masks validation loss



(b) Bounding Boxes validation loss



(c) Total validation loss

Figure 5.4: Thermal channel training behaviour.

The number of epochs, training steps and validation steps were altered, but did not modify its overall tendency. It shows poor results and starts overfitting early when training with a higher learning rate, equally to the Near-IR channel mentioned in 2.4.2. When trained with a higher number of epochs, the total validation loss never crossed the 0.625. This was the lowest loss value obtained and is displayed in the above graphs. The training

processes with maximum of 120 epochs took an average of 12 hours to conclude.

Nonetheless, the Thermal band was the one that presented the best training results with lower losses, which in turn reveals the best detection and final results. This is mainly due to the fact that the Thermal sensor measures, essentially, temperature. Its data is displayed in shades of grey, with higher temperature areas being represented with higher pixel intensity, almost white, and lower temperatures with darker grey. This simplified data representation allows the network to train and run through the validation set without having as much factors to consider such as objects shapes or reflections, and focusing mainly on pixels intensity.

5.2.4 Training Results Analysis

The table 5.2 bellow presents a good overview on these results by showing all loss values for each weights file selected.

Table 5.2: Loss values for each channel.

Loss Values	Red Edge	Near-IR	Thermal
val_mrcnn_bbox_loss	0.4493	0.3499	0.1769
val_mrcnn_class_loss	0.0730	0.0673	0.0336
val_mrcnn_mask_loss	0.4156	0.3547	0.2295
val_rpn_bbox_loss	0.5496	0.4986	0.1776
val_rpn_class_loss	0.1277	0.0998	0.0074
Total Validation Loss	**1.6152**	**1.3703**	**0.6250**

By examining the table, it is evident that the class loss values are the ones with the smallest impact on the total loss. This is common on the three channels and is the result of training a single class network. In these desired conditions, the objects can only be classified as fire or background, leaving a smaller chance of failure in class detection.

The Red Edge channel, besides showing the highest loss values, was the only channel with a bounding box loss higher than mask loss. This fact helps proving that, with this channel, the network has a considerable difficulty defining the boundaries of fire instances, essentially, where it starts and where it stops. This, along with what was already discussed in 5.2.2 proves that this band's data is the least suited to be used in the models inference process.

The Near-IR channel network presents loss values comparable to a single class network trained with basic RGB imagery [75]. Although it is not a bad outcome, it does not entirely justify its existence when a simpler network can achieve similar results. This makes Near-IR data good for future usage but not ideal with the network in its current situation.

It is clear, considering the table, that even though the Near Infra Red presented considerable better results comparing to the Red Edge, the Thermal band was by far the one with the best results, with the smallest values in every loss metric. It presents a total

validation loss of less than half of the fourth channel and thus, is the most suited for future real-life applications.

5.3 Inference Evaluation

Even though it is essential to evaluate the training process and compare loss values between the three used channels, a more practical approach is also crucial to assess the implemented model performance under real-life scenarios. As such, in order to evaluate this model's accuracy, efficiency and robustness, the Mean Average Precision (mAP) with Intersection over Union (IoU) of 0.5 was computed.

The mAP method is the result of the mean of all images' Average Precision values which depend directly on precision and recall metrics. Precision measures the amount of predictions that are correct from all predictions and is calculated by:

$$Precision = \frac{TP}{TP + FP} \tag{5.1}$$

Recall, on the other hand, measures the amount of instances that were correctly detected and is calculated by:

$$Recall = \frac{TP}{TP + FN} \tag{5.2}$$

The AP for an image corresponds to the area under the precision-recall curve and is ideally 1, with all recall values corresponding only to precision values of 1[76].

However, since object detection is not a binary problem and its success cannot be measured simply by checking if the predicted detection matches the ground truth, the IoU evaluation metric is necessary. The IoU is intended to attenuate the evaluation rigour, allowing to consider detections with small deviations as correct. This metric is calculated with:

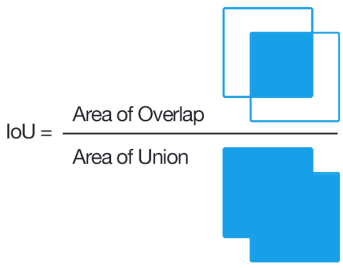


Figure 5.5: IoU calculation [77].

If the value obtained is greater than the defined threshold, it is considered as a success, a True Positive, but if, on the other hand, is inferior, it is considered a False Positive. For these results evaluation, the threshold value was set to 0.5 which is the default value in most applications and is enough for object detection in images taken from distances between 10 and 120 meters.

5.3.1 Test set Structuring

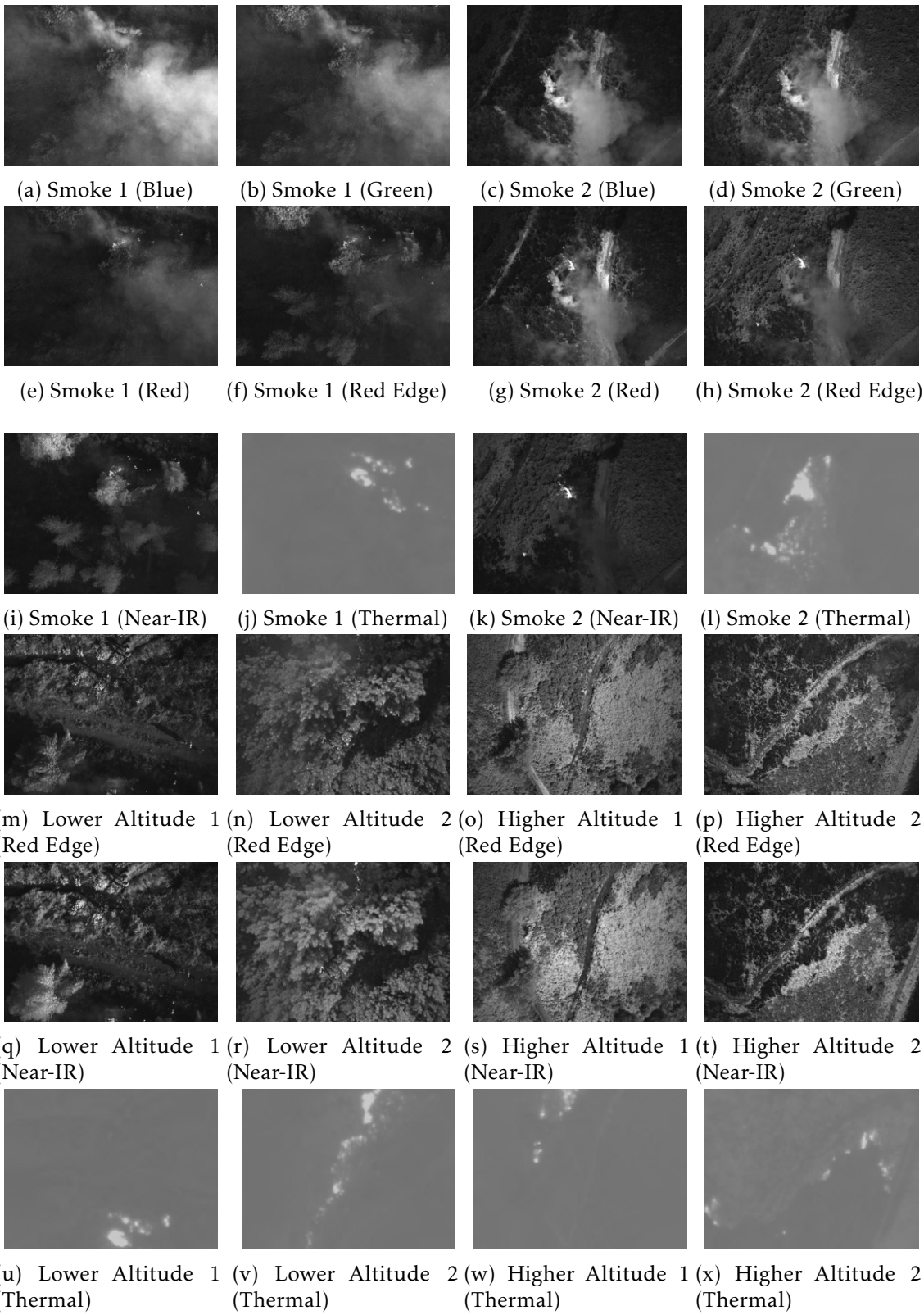
To apply the mAP metric, a test set was selected, containing a group of 180 images also captured with the same camera mentioned in 3 and 4.2.1. Unfortunately, due to the Covid-19 pandemic and the time these tests were performed, it was not possible to gather more data to compose a new data set to be used as test set. As such, a group of images from the validation set were selected to be part of the test set. From this testing group, similarly to the training set, three sub sets were acquired, each one having a different channel for the testing images set, to test the implemented model for different bands in equal conditions. The images were selected based on their backgrounds, unique characteristics and differences from each others, in order to obtain the most diverse and representative test set. The data sets can be organised by environment types to easily demonstrate their diversity and simplify the results evaluation process, by covering the greatest variety of examples possible. Some examples of these images organised by their characteristics(belonging to both training and validation sets) are presented in figure 5.6. Their presence within the data set is also presented in table 5.3.

Table 5.3: Amount of images from each category within the data set.

Categories	**(%)**
Smoke	41
Lower Altitude	59
Higher Altitude	41

The first two images (from figure 5.6a to 5.6l) are referring to scenarios where the air is presented with an extreme amount of smoke. These cases were the only ones where all six bands were displayed, since the RGB channels are essential to help visualise these specific circumstances. When comparing these channels' data with the other three, it can be observed that the RGB are a lot more sensitive to smoke, which in turn results in a greater difficulty in distinguishing fire instances. This is a very important point that highlights and demonstrates why the Red Edge, Near-IR and Thermal bands are considered the better option for a fire classification and localisation model. Even if using the RGB bands for fire detection produced better results, it would only happen under specific conditions where smoke was not present in the camera's field of view.

The second half of the figure 5.6 presents four images, exposing the altitude differences existing in the data sets. Some fires were captured at lower altitudes(between 15 and 20 meters) like the first two images (figures 5.6m, 5.6q, 5.6u, 5.6n, 5.6r and 5.6v) and others at higher altitudes(between 100 and 120 meters), like the other two (figures 5.6o, 5.6s, 5.6w, 5.6p, 5.6t and 5.6x), to try and identify the most suitable height to collect data. This ideally matches the height with which the images with the best results were captured. Nonetheless, this matter needs to be rigorously concluded, taking into account all other factors that may highly influence the obtained results.



(a) Smoke 1 (Blue) (b) Smoke 1 (Green) (c) Smoke 2 (Blue) (d) Smoke 2 (Green)

(e) Smoke 1 (Red) (f) Smoke 1 (Red Edge) (g) Smoke 2 (Red) (h) Smoke 2 (Red Edge)

(i) Smoke 1 (Near-IR) (j) Smoke 1 (Thermal) (k) Smoke 2 (Near-IR) (l) Smoke 2 (Thermal)

(m) Lower Altitude 1 (Red Edge) (n) Lower Altitude 2 (Red Edge) (o) Higher Altitude 1 (Red Edge) (p) Higher Altitude 2 (Red Edge)

(q) Lower Altitude 1 (Near-IR) (r) Lower Altitude 2 (Near-IR) (s) Higher Altitude 1 (Near-IR) (t) Higher Altitude 2 (Near-IR)

(u) Lower Altitude 1 (Thermal) (v) Lower Altitude 2 (Thermal) (w) Higher Altitude 1 (Thermal) (x) Higher Altitude 2 (Thermal)

Figure 5.6: Data set examples divided in three main categories, images with high amounts of smoke, images captured from lower altitudes and images captured from higher altitudes.

Another important factor that is also extremely important is the vegetation density. Unfortunately, there are no images with truly low densities within the gathered data sets and the figure 5.6 is not divided by those categories. However, there are multiple cases like the first smoke image (figures 5.6a, 5.6b, 5.6e, 5.6f, 5.6i and 5.6j), the second lower altitude image (figures 5.6n, 5.6r and 5.6v) and the first higher altitude (figures 5.6o, 5.6s and 5.6w), that present thicker and larger trees compared to the other images. These types of trees, with larger and more dense canopies are usually tall enough to leave room for low fire to burn, without being visible from a top perspective. It is also important to determine if in these cases the model is also able to detect burning instances that are occurring below these canopies.

5.3.2 Inference Results

The inference tests that follow were performed in Google CoLab notebook with detection_min_confidence of 0.8. For each subset, all the different images were given as input to the model and the number of True Positives (TPs), True Negativess (TNs), False Positives (FPs) and False Negatives (FNs) were registered. With these numbers, a list of precisions and a list of recalls were determined for each image and the AP was computed by compute_ap in utils.py. The mean of all the AP values was obtained and is displayed, for each trained band, in the table 5.4 below.

Table 5.4: Mean Average Precision values with IoU=0.5 and Inference Times for each band.

	Red Edge	**Near-IR**	**Thermal**
mAP(%) per Category			
Lower Altitude Category	29.2	44.9	91.8
Higher Altitude Category	13.2	37.4	92.3
Smoke Category	26.1	42.7	91.2
Total mAP(%)	**25.3**	**43.2**	**91.7**
Inference Time(s)	11.99	11.72	10.68

Before drawing conclusions about the different bands results, it is possible to interpret some more general information concerning multispectral fire detection, by observing the table. The Red Edge and Near Infra Red, both bands affected by visual changes (unlike the Thermal band), show the same behaviour in regards to smoke and altitude changes. Even with only 41% of images containing substantial amounts of smoke, both bands present similar mAP values in detections at high altitudes and in overall detections (total mAP value). This indicates that the presence of smoke does not change the behaviour of the Red Edge and Near-IR trained networks and that these models are suitable to perform under these circumstances. In regards to the UAV's altitude, both bands revealed better performances (mAP value closer to the overall mAP value) at lower altitudes.

When studying the table, it is clear that the mAP computed values match with the conclusions drawn from the previous section. The Near-IR band is also superior to the Red Edge band by almost 20%. The Red Edge presents a really low value of 25.3% and its discrepancy from the Near-IR band reinforces the fact that this spectral band data is not, only by itself, suited to be used in the final system.

The mAP value obtained by the Near-IR band is an average value that is not high enough compared to the average mAP of some object detector networks, presented in figure 5.1. For this case, a single class model, even though a precision value greater than 80% would be optimal, a value slightly greater than 50% would also be acceptable. There are lots of cases where the burning areas are not intense enough or may not be even visible, which causes the model to present a higher number of False Negatives. However, with the mAP value of 43.2%, there are still a lot of missed instances that should not occur as well as False Positives that can definitely be prevented. The Near-IR band based model can be further improved with future trainings but at the moment, may not be suited to be used in a system operating in a real life scenario.

The Thermal band presents a vastly superior mAP value of 91.7%, proving to be excellent in high temperature areas detection. Similar to the training results, this band based model has results far more positive than the others, proving it does not need to continue training and can be currently implemented in the system.

Despite being the best metric to quantify object detection quality, it is important to understand that there are always some factors that the mAP may have difficulty accounting for. Since this metric depends mainly on precision and accuracy values and these always consider the existence of True Positives, all images that have no instances (TPs) are not considered. This can be a real issue if there are a lot of false positives in these images, ending up being ignored.

Another example is the fact that when the detection misses the IoU threshold, it is identified not only a False Positive but also a False Negative, decreasing both precision and recall. This is particularly problematic in images with large numbers of small instances that may be more easily missed.

In addition to these smaller issues, there is a main problem, directly related with the type of data used in this dissertation, that influenced the mAP values to be lower than expected for all three bands. When detecting fire areas, the human eye can differentiate isolated burning areas, considering each area as an instance. However, the network besides doing that, when detecting inside each area highly different levels of intensity, considers it as different instances. These smaller instances inside the original instance, do not have enough area overlapping the true areas and are considered as False Positives. Some examples are presented in the next section, in figure 5.7. It is important to mention that, despite contributing for worse mAP results, these occurrences have no negative impact in the models results, only implying different intensity levels within the detected areas.

The inference times presented in the table were obtained in a Intel Core i7 4710HQ

2.50GHz Laptop. The CoLab resources fluctuate often and do not guarantee a constant behaviour, which would be the ideal conditions to test a process running time. Without an access to a GPU, the inference times obtained were not the desired, but can still be analysed. These values also imply the model's capacity to run in a real time scenario when having access to more suitable hardware conditions.

The inference times displayed represent the time between the input of the multispectral images into the model and the saving of the model's output. As it is shown in the the table 5.4, the thermal inference process presents a slightly faster response compared to the others bands. This can justified by the thermal images inferior dimensions. Their dimensions, despite increased beforehand, are still half the size of the other bands images, which is reflected in the Mask R-CNN's detection time.

5.3.3 Practical Examples

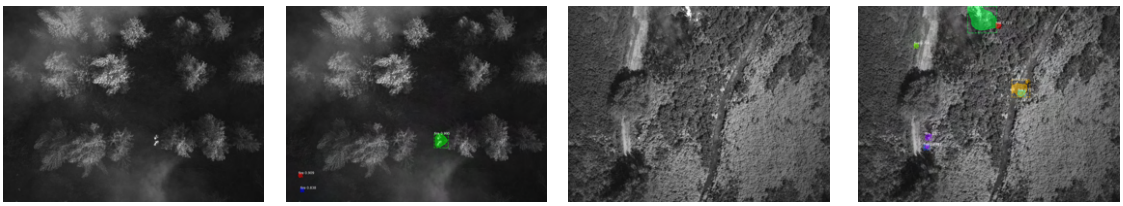
Some examples of the final models output operating with each of the spectral bands individually may be observed in figure 5.7. These cases were obtained from the test set and were selected based on their diversity, representing all different types of images discussed in the data set examples. Image 1 (figures 5.7a, 5.7e and 5.7i) has an excessive amount of smoke, image 2 (figures 5.7c, 5.7g and 5.7k) was captured from a higher altitude, image 3 (figures 5.7m, 5.7q and 5.7u) and 4 (figures 5.7o, 5.7s and 5.7w) from a lower altitude and images 1, 3 and 4 have all larger trees with dense canopies.

These examples also cover the necessary objects and situations to objectively illustrate the inference accuracy of each band, as well as their pros and cons. The inference process for these cases was performed with detection_min_confidence set to 0.7 opposed to the 0.8 used in the mAP computation. Even though the network presents better results with the higher value, 0.7 was chosen for this purpose to emphasise some common errors that are more frequent and can be all more easily addressed in a smaller group of images.

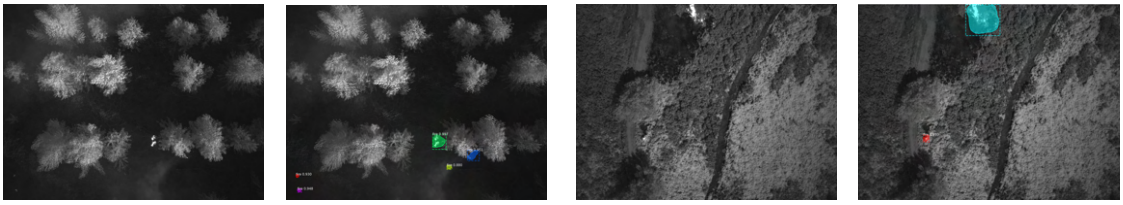
5.3.3.1 Red Edge band

Images 1 (figures 5.7b and 5.7f) and 2 (figures 5.7d and 5.7h) are good examples of the fifth channel quality relatively to the fourth. In image 1, all performed detections are correct, but the Near-IR channel network manages to detect more instances than the Red Edge network. The smoke is a lot more visible in the Red Edge, covering some possible less intense instances. In this and third image (figure 5.7n), the instances in areas that correspond to the trees more central and dense points (visible through the thermal figures 5.7j and 5.7v) are either not completely localised or not detected. In image 2 (higher altitude), with the help of the Thermal image, it is visible that the Red Edge network, despite having a higher number of detected instances than the Near-IR, presents mostly False Positives.

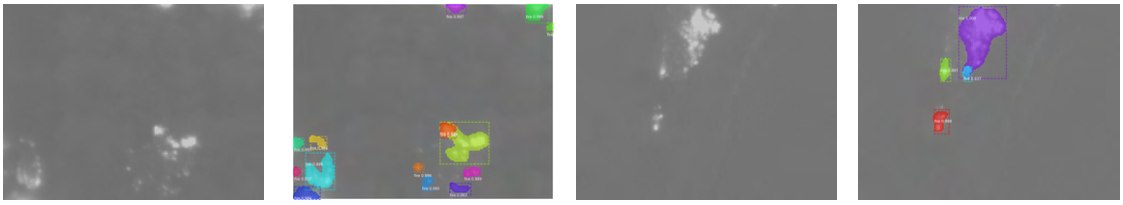
Observing image 4 (figure 5.7p) a common issue is displayed regarding the Red Edge associated detection. As mentioned before in 5.2.4, the bounding boxes are regularly



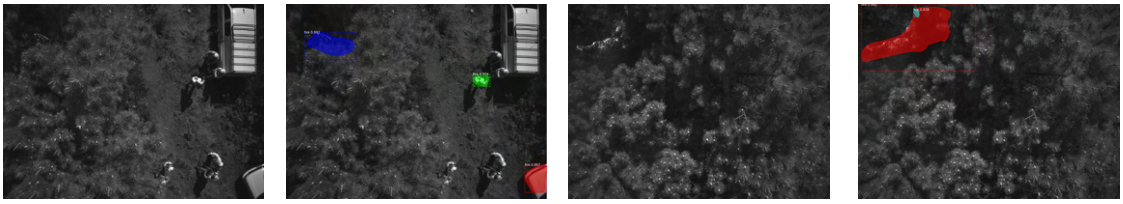
(a) Image 1 (Red Edge) (b) Image 1 (Red Edge) inference (c) Image 2 (Red Edge) (d) Image 2 (Red Edge) inference



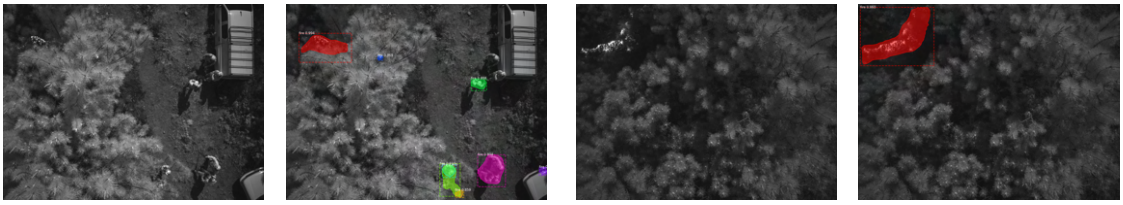
(e) Image 1 (Near-IR) (f) Image 1 (Near-IR) inference (g) Image 2 (Near-IR) (h) Image 2 (Near-IR) inference



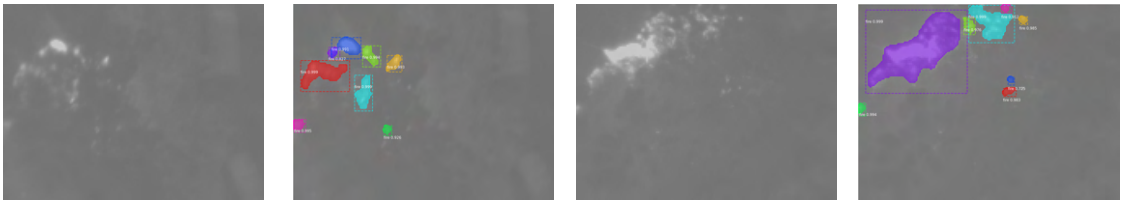
(i) Image 1 (Thermal) (j) Image 1 (Thermal) inference (k) Image 2 (Thermal) (l) Image 2 (Thermal) inference



(m) Image 3 (Red Edge) (n) Image 3 (Red Edge) inference (o) Image 4 (Red Edge) (p) Image 4 (Red Edge) inference



(q) Image 3 (Near-IR) (r) Image 3 (Near-IR) inference (s) Image 4 (Near-IR) (t) Image 4 (Near-IR) inference



(u) Image 3 (Thermal) (v) Image 3 (Thermal) inference (w) Image 4 (Thermal) (x) Image 4 (Thermal) inference

Figure 5.7: Inference examples of 4 different images.

poorly defined, as is the case in the red instance that is displayed. Compared with the other two channels, the box extends too far to the right side, almost reaching the centre of the image.

5.3.3.2 Near Infra Red band

With image 1 (figure 5.7f), it is visible the advantage that the Near-IR has when dealing with a smoke environment, almost avoiding it entirely. This model also presents a better precision compared to the fourth channel when presented with bigger and more dense tree canopies. Despite being better and more suited for these cases, it still does not present desired results, as it is displayed by the trees in figures 1 and 3. However, when examining image 3 (figure 5.7r), it is noticeable the main problems that the Near-IR trained model presents. Compared with the Red Edge band (figure 5.7n), it detects the fire instances with a better accuracy and precision, but also raises a larger amount of False Positives. Most of the firemen are considered fire with a high confidence level as well as the reflection on the car on the bottom right corner. This is an undeniable issue that may be detrimental when performing under real-life circumstances, since these elements are regularly on scene and will be surely presented to the model in the future. However, this fact can be mitigated by later human supervision of detected ignitions. A system that could filter human instances could also be implemented in the future.

5.3.3.3 Thermal band

All four inference examples (figures 5.7j, 5.7l, 5.7v and 5.7x) help demonstrate the extreme accuracy and precision the Thermal band presents on detecting fire occurrences and higher temperature areas. As mentioned before, and verified with some examples, this is the channel with which the best results were obtained. As expected, this band inference is not affected by the image types mentioned and displayed before in figure 5.6, meaning its performance is the same presented with smoke, different types of vegetation and at different capturing altitudes.

It is visible a larger quantity of instances detected, with a great amount of these corresponding to areas where flames are not discernible in the other channels. Some examples are on images 1 (figure 5.7j) and 4 (figure 5.7x). Part of these extra detected instances correspond to already identified areas that occasionally even overlap each other like the orange and yellow masks in image 1. This is a consequence of the different pixels intensity since a big area with an overall high temperature may have within it smaller hotter areas. As mentioned before in this section, the network may sometimes detect a bigger area as one instance and the smaller more intense areas inside as others, which in turn, helps justifying the loss value present in 5.2.3. Despite adding to the total loss value, this issue is not viewed as a downside for the implemented model final result as it only presents False Positives that are considered correct to the user.

5.3.4 Inference Results Analysis

Considering both computed precision values and analysed output examples, and also taking the training results from section 5.2.4 into account, it is possible to draw a significant conclusion on the models inference results.

The results obtained for the networks trained with each of the three bands individually reveal that in order to achieve the completely desired outcome, with the most robust and reliable responses possible, it is necessary to combine the results from different spectral detections. A system based solely on RGB imagery or multispectral imagery or thermal imagery will not cover nor recognise every important feature that needs to be captured and comprehended by the detection network. The RGB detection[75] has an evident problem in regards to smoke, without an effective method to overcome it. The multispectral Red Edge and Near-IR on the other hand, tackle this issue positively but still do not expose all instances in a recognisable way for the network (mainly the ones covered by more dense vegetation). The thermal detection manages to localise all high temperature areas, but this data can only be fully utilised when compared with other visual bands, otherwise it is impossible to conclude which areas do actually correspond to fire instances. Nonetheless, it is crucial to cover in detail these different detection results for each band and understand the advantages and disadvantages that the Red Edge, Near-IR and Thermal bands present.

The Red Edge band presented the worse results overall with the lowest mAP value, a very high amount of False Positives and a limited capacity to detect minor fire instances. From the three used bands, this is the most subjective to smoke and despite its usual darker shades, it does not present fire instances with brighter contrasts, which the detection model would benefit. Its data also does not introduce any feature that makes it favourable over the Near-IR band. This reinforces that this band will not be used individually in the model or in future fire detectors' implementations.

The Near Infra Red band on the other hand, demonstrated to be suitable in the fire detection field with a considerably higher precision level than the Red Edge. Although not exceptional, its precision level is similar to that of some applications where localisation is not required to be extremely precise. The fifth band manages to capture the field with almost zero smoke signs, which helps acquiring a clearer view, allowing in turn for a sharper detection. However, it showed clear signs of vulnerability in some of the output examples discussed in the previous subsection and fragility with a mAP value still lower than desired. It is not able to always detect all equal intensity burning areas and shows clear difficulties when faced with objects outside forest and natural environments such as vehicles and people. This is, as already mentioned in this section, due to a mildly specific training set that was not as big and varied as desired and did not have enough diverse objects in its image's background. Even though it is able to tackle the smoke problem very efficiently, when considering images where smoke is not present, a Near Infra Red band based model cannot currently be considered an improvement compared to a more

standard RGB fire detector.

The thermal band presented the best overall results, having a mAP with double the value of the Near-IR band and all displayed example instances accurately classified and localised. It currently demonstrates an excellent detection performance suitable for real-life applications. As sections 4.3.1 and 4.3.2 demonstrated, a model working only based on this band does not produce a comprehensible output for the user to interpret. With access to other bands data, it is possible to overlap (section 4.3.1) the produced masks in a way the user can assimilate. This method presents the user with the locations for all high temperature areas, which despite being extremely valuable for different operations, does not bring any practical advantages by itself without any sort of selection. Therefore, in order to benefit properly from this thermal model results and achieve more elaborated outputs (rekindle detection for example), it is essential to include as well an accurate fire detection outside the thermal band.

Both Near-IR and Red Edge results showed an overall better detection performance when confronted with lower altitude images (table 5.4). These images were presented in most cases with more assertive fire localisation and higher confidence values. The contrast found around most fire instances is not as apparent at higher altitudes and both bands displayed worse results with this type of data. This indicates that the model performs better with lower altitude images and that when used in a real-life scenario, it would be ideal to have the UAV flying and capturing images at those altitudes.

Relatively to the trees' dimensions, it was determined that when under very large and dense canopies, the fire instances in both Red Edge and Near-IR images present more difficulties to be detected than in other areas. The high leaf and branches density can completely cover the area below a tree, preventing any sign of visible fire to be captured from a top view. The Near-IR detection presented better results since this band's data presents the fire instances with brighter contrasts, but did still perform poorly under these types of situations. A method to allow the UAV to collect data from these covered regions needs to be implemented in the future in order to have a more solid and dependable fire detector.

Further adjustments need to be applied to the visual data detection, mainly to the Near-IR band detection. The network requires a more complete and exhaustive training with wider data sets containing data from the most diverse environments possible. To guarantee the best improvement, the data gathered should be captured at lower altitudes and also include a lot of examples containing objects that may be recurrent in wildfire scenarios, such as fire trucks like the ones presented in the examples from the previous section. The Near Infra Red model did present a lot of False Positives related to these objects that can be prevented, resulting in much more assertive and reliable responses. This improvement is also crucial to continuing developing and upgrading the rekindle classification and localisation. The Near-IR is, at the moment, essential for this method and is required to perform with the highest levels of accuracy and precision.

Chapter 6

Conclusions and Future Work

6.1 Conclusions

In this document, a problem regarding wildland fire detection was addressed and a solution, in order to approach the situation at hand, was executed. The proposed solution, inserted on a UAV monitoring system, intended the development of a module capable of receiving real time images captured by a multispectral camera and detecting the presence of fire or possible rekindles.

For a better understanding on the different technologies necessary for the correct implementation of the solution proposed and more detailed information on the work already developed around the field in question, a State of the Art was necessary to be reviewed. This section covered different types of image acquisition that could be considered more relevant for fire detection and different machine learning classifiers commonly used in image classification. Convolutional Neural Networks were particularly explored in this dissertation's State of the Art as well as different frameworks based on these networks to select the ones more appropriated to the model implementation.

Next, a model intended to solve the problem was proposed and carefully explained. In chapter 4 this model's implementation is thoroughly explained. The model was implemented to receive one of three spectral bands image, Red Edge, Near Infra Red and Thermal, detect the exact pixel location where fire areas are located, overlap these with coloured masks and save the output image. The Mask R-CNN object detection framework was selected to incorporate this model and perform the detection phase. With its tuned parameters, it was trained for the three different bands and their inference results minutely compared in chapter 5.

The best results were obtained with the thermal band where most of the ground truth areas were detected with extreme accuracy and precision. With the immensely positive

outcomes displayed in the results section, it can be concluded that a method capable of detecting extreme high temperature areas was successfully implemented. The Near Infra Red band presented an high precision when detecting visible burning areas, even in the presence of smoke, but not high enough to be considered currently practicable. Despite its promising behaviour, the Near-IR band detection model still needs more extensive training with a more complete and diverse data set to present slightly better results and be viable in real-life scenarios. It was also concluded that to achieve a more consistent system with stable results, it is essential to combine various bands and their model's results. In doing so, it is possible do acquire a more complete system able to detect more precisely existent fire instances making use of the different bands advantages.

An approach to rekindle detection was also included in this dissertation, with the successful detection of areas not considered fire instances but with extreme high temperatures. This first attempt presented a positive outcome that shall be further explored and developed in the future.

In conclusion, the multispectral fire detector with the use of the Mask R-CNN framework presented good overall results, indicating a promising outcome for future implementations. These obtained results do not represent an ideal final solution but demonstrate the model's potential in the wildfire fighting field, reinforcing the importance of continuing to study and improve it.

6.2 Future Work

Despite this dissertation presenting satisfying results for real time fire detection, the presented solution might not yet be prepared for real life usage. There are improvements that are required for a steadier and more robust behaviour and some features that should be considered and added to bring better detailed results and more desired features. Possible procedures to be implemented in the future for further improving the system are presented here:

- The data used in the training process was recovered from seven different groups of aerial images, some of these with similar surroundings. This model would benefit immensely with access to a larger set of images gathered from the most diverse environments possible. To collect this extra data, more prescribed fires must be arranged, which, due to the current pandemic restrictions and weather conditions, will not be possible before Spring 2021.

  It would also be extremely important to gather a significant amount of data for test purposes only. The inference tests performed for this dissertation were adequate but in order to guarantee completely unbiased test results, the test set should be one that the model never had contact with before.

- Due to the results obtained and displayed in chapter 5, the Near Infra Red and Thermal bands were the most explored. Despite this, the results do not implicate that the best approach to implement the desired system is exclusively based on these two bands data. Studies should be performed to understand if combining and making use of both Near-IR and Red Edge bands could introduce more information and be beneficial to the overall system. This could prove to be extremely relevant and eventually replace the Near-IR band detection as the visible fire detector.

  It would also be vital to understand with additional studies if the detections performed with the Near-IR model can, in the future, correspond to all visible fire areas. This is an extremely important aspect that if assured, will improve the rekindle detection and guarantee much more reliable and robust results.

- The implemented thermal detection, that has a major role on this dissertation, is implemented making use of an alignment method that may not be suited for future captured images. A different approach must be realised to acquire a more generalised image alignment method, capable of producing more cohesive and robust results.

- Despite showing good detection results, even for low intensity fires, the presented model does not present a final solution for detecting and differentiating rekindle areas. The detection of high temperature areas that do not present fire characteristics was successfully performed. However, this process does not guarantee that the detected areas are all future rekindle areas and currently cannot be used in real life scenarios. Further studies need to be performed in order to recognise from these areas, the ones that are more propitious to initiate fire outbreaks.

- As it was mentioned in the Experimental Results section 5.3.4, the model implemented reveals some difficulties localising certain fire instances that are covered by larger trees for example. One way to counter this issue is by having the UAV, that is responsible for collecting the data, moving according to its surroundings. It would be ideal to implement a model that, with access to the 3D data, was able to recognise the best areas to collect the most complete ground images and move the UAV to those precise locations.

Bibliography

[1] A. P. Pacheco, J. Claro, and T. Oliveira. "Rekindle dynamics: Validating the pressure on wildland fire suppression resources and implications for fire management in portugal." In: *WIT Transactions on Ecology and the Environment* 158 (2012), pp. 225–236. ISSN: 17433541. DOI: 10.2495/FIVA120191.

[2] A. C. Meira Castro, A. Nunes, A. Sousa, and L. Lourenço. "Mapping the causes of forest fires in Portugal by clustering analysis." In: *Geosciences (Switzerland)* 10.2 (2020), pp. 7–11. ISSN: 20763263. DOI: 10.3390/geosciences10020053.

[3] J. Su, C. Liu, X. Hu, X. Xu, L. Guo, and W. H. Chen. "Spatio-temporal monitoring of wheat yellow rust using UAV multispectral imagery." In: *Computers and Electronics in Agriculture* 167.July (2019), p. 105035. ISSN: 01681699. DOI: 10.1016/j.compag.2019.105035. URL: https://doi.org/10.1016/j.compag.2019.105035.

[4] X. Wang, H. Sun, Y. Long, L. Zheng, H. Liu, and M. Li. "Development of Visualization System for Agricultural UAV Crop Growth Information Collection." In: *IFAC-PapersOnLine* 51.17 (2018), pp. 631–636. ISSN: 24058963. DOI: 10.1016/j.ifacol.2018.08.126. URL: https://doi.org/10.1016/j.ifacol.2018.08.126.

[5] C. R. Pennypacker, M. K. Jakubowski, M. Kelly, M. Lampton, C. Schmidt, S. Stephens, and R. Tripp. "FUEGO - Fire urgency estimator in geosynchronous orbit - A proposed early-warning fire detection system." In: *Remote Sensing* 5.10 (2013), pp. 5173–5192. ISSN: 20724292. DOI: 10.3390/rs5105173.

[6] H. Kaur and S. K. Sood. "Journal of Network and Computer Applications Fog-assisted IoT-enabled scalable network infrastructure for wildfire surveillance." In: *Journal of Network and Computer Applications* 144.June (2019), pp. 171–183. ISSN: 1084-8045. DOI: 10.1016/j.jnca.2019.07.005. URL: https://doi.org/10.1016/j.jnca.2019.07.005.

[7] B. U. Töreyin, Y. Dedeoğlu, U. Güdükbay, and A. E. Çetin. "Computer vision based method for real-time fire and flame detection." In: *Pattern Recognition Letters* 27.1 (2006), pp. 49–58. ISSN: 01678655. DOI: 10.1016/j.patrec.2005.06.015.

[8] K. Poobalan and S.-c. Liew. "Fire detection algorithm using image processing techniques." In: *Proceedings of the 3rd International Conference on Artificial Intelligence and Computer Science (AICS2015)* December (2015), pp. 12–13. URL: https://www.researchgate.net/publication/285580944.

[9] T. Celik. "Fast and efficient method for fire detection using image processing." In: *ETRI Journal* 32.6 (2010), pp. 881–890. ISSN: 12256463. DOI: 10.4218/etrij.10.0109.0695.

[10] T. S. Huang and C. E. Vandoni. "Computer Vision: Evolution And Promise, CERN School of computing." In: *CERN EUROPEAN ORGANIZATION FOR NUCLEAR RESEARCH REPORT- CERN, CERN School of computing*. 8. CERN; 1996, pp. 21–26. ISBN: 9290830956. URL: https://www.tib.eu/de/suchen/id/BLCP\%3ACN017866441.

[11] L. Roberts. *Machine Perception of Three-Dimensional Solids*. Vol. 7. 3. 1963, pp. 133–146. DOI: 10.1080/17470215508416686.

[12] H. Mo and C. Luo. "Robot Indoor Navigation Based on Computer Vision and Machine Learning." In: 9713.June 2016 (2016), pp. 461–469. DOI: 10.1007/978-3-319-41009-8. URL: https://doi.org/10.1007/978-3-319-41009-8.

[13] S. Jong and J Clevers. "Basics of Remote Sensing." In: July (2007), pp. 211–212. DOI: 10.1007/978-1-4020-2560-0.

[14] Y. Lu and T. Javidi. "Efficient Object Detection for High Resolution Images." In: (2015), pp. 1091–1098. DOI: 10.1109/ALLERTON.2015.7447130.

[15] H. Tayara and K. T. Chong. "Object Detection in Very High-Resolution Aerial Images Using One-Stage Densely Connected Feature Pyramid Network." In: (2018), pp. 1–18. DOI: 10.3390/s18103341.

[16] V. Ruzicka and F. Franchetti. "Fast and accurate object detection in high resolution 4K and 8K video using GPUs." In: *2018 IEEE High Performance extreme Computing Conference (HPEC)* (2018). DOI: 10.1109/hpec.2018.8547574. URL: http://dx.doi.org/10.1109/HPEC.2018.8547574.

[17] *Sensors 101: The Basics of LiDAR, Thermal, Hyperspectral, and Multispectral Technology*. URL: https://www.precisionhawk.com/blog/media/topic/sensors-101-basics-lidar-thermal-hyperspectral-multispectral-technology (visited on 01/24/2020).

[18] *Drone-based Multispectral Sensing: what to know*. URL: https://www.precisionhawk.com/blog/media/topic/drone-based-multispectral-sensing-what-to-know (visited on 01/31/2020).

[19] J. D. Burnett and M. G. Wing. "A low-cost near-infrared digital camera for fire detection and monitoring." In: *International Journal of Remote Sensing* 39.3 (2018), pp. 741–753. ISSN: 13665901. DOI: 10.1080/01431161.2017.1385109.

[20] C. Burke, S. Wich, K. Kusin, O. McAree, M. Harrison, B. Ripoll, Y. Ermiasi, M. Mulero-Pázmány, and S. Longmore. "Thermal-Drones as a Safe and Reliable Method for Detecting Subterranean Peat Fires." In: *Drones* 3.1 (2019), p. 23. ISSN: 2504-446X. DOI: 10.3390/drones3010023.

[21] Y. Li, A. Vodacek, R. L. Kremens, A. Ononye, and C. Tang. "A hybrid contextual approach to wildland fire detection using multispectral imagery." In: *IEEE Transactions on Geoscience and Remote Sensing* 43.9 (2005), pp. 2115–2125. ISSN: 01962892. DOI: 10.1109/TGRS.2005.853935.

[22] M. J. Sousa, A. Moutinho, and M. Almeida. "Classification of potential fire outbreaks: A fuzzy modeling approach based on thermal images." In: *Expert Systems with Applications* 129 (2019), pp. 216–232. ISSN: 09574174. DOI: 10.1016/j.eswa.2019.03.030. URL: https://doi.org/10.1016/j.eswa.2019.03.030.

[23] L. Deng, Z. Mao, X. Li, Z. Hu, F. Duan, and Y. Yan. "UAV-based multispectral remote sensing for precision agriculture: A comparison between different cameras." In: *ISPRS Journal of Photogrammetry and Remote Sensing* 146.September (2018), pp. 124–136. ISSN: 09242716. DOI: 10.1016/j.isprsjprs.2018.09.008.

[24] L. Rey-Barroso, F. J. Burgos-Fernández, X. Delpueyo, M. Ares, S. Royo, J. Malvehy, S. Puig, and M. Vilaseca. "Visible and extended near-infrared multispectral imaging for skin cancer diagnosis." In: *Sensors (Switzerland)* 18.5 (2018), pp. 1–15. ISSN: 14248220. DOI: 10.3390/s18051441.

[25] T. A. Carrino, A. P. Crósta, C. L. B. Toledo, and A. M. Silva. "Hyperspectral remote sensing applied to mineral exploration in southern Peru: A multiple data integration approach in the Chapi Chiara gold prospect." In: *International Journal of Applied Earth Observation and Geoinformation* 64.March (2018), pp. 287–300. ISSN: 1872826X. DOI: 10.1016/j.jag.2017.05.004.

[26] G. J. Edelman, E. Gaston, T. G. van Leeuwen, P. J. Cullen, and M. C. Aalders. "Hyperspectral imaging for non-contact analysis of forensic traces." In: *Forensic Science International* 223.1-3 (2012), pp. 28–39. ISSN: 03790738. DOI: 10.1016/j.forsciint.2012.09.012.

[27] T. Sankey, J. Donager, J. McVay, and J. B. Sankey. "UAV lidar and hyperspectral fusion for forest monitoring in the southwestern USA." In: *Remote Sensing of Environment* 195 (2017), pp. 30–43. ISSN: 00344257. DOI: 10.1016/j.rse.2017.04.007. URL: http://dx.doi.org/10.1016/j.rse.2017.04.007.

[28] S. Veraverbeke, P. Dennison, I. Gitas, G. Hulley, O. Kalashnikova, T. Katagis, L. Kuai, R. Meng, D. Roberts, and N. Stavros. "Hyperspectral remote sensing of fire: State-of-the-art and future perspectives." In: *Remote Sensing of Environment* 216.November 2017 (2018), pp. 105–121. ISSN: 00344257. DOI: 10.1016/j.rse.2018.06.020. URL: https://doi.org/10.1016/j.rse.2018.06.020.

[29] S. B. Kotsiantis, I. D. Zaharakis, and P. E. Pintelas. "Machine learning: A review of classification and combining techniques." In: *Artificial Intelligence Review* 26.3 (2006), pp. 159–190. ISSN: 02692821. DOI: 10.1007/s10462-007-9052-3.

[30] *Understanding Random Forest - Towards Data Science*. URL: https://towardsdatascience.com/understanding-random-forest-58381e0602d2 (visited on 01/31/2020).

[31] *Decision tree-based ensemble methods - TensorFlow Machine Learning Projects*. URL: https://subscription.packtpub.com/book/big_data_and_business_intelligence/9781789132212/2/ch02lvl1sec21/decision-tree-based-ensemble-methods (visited on 02/02/2020).

[32] *Why Random Forest is My Favorite Machine Learning Model*. URL: https://towardsdatascience.com/why-random-forest-is-my-favorite-machine-learning-model-b97651fa3706 (visited on 01/31/2020).

[33] O. Kim and D.-J. Kang. "Fire detection system using random forest classification for image sequences of complex background." In: *Optical Engineering* 52.6 (2013), p. 067202. ISSN: 0091-3286. DOI: 10.1117/1.oe.52.6.067202.

[34] W. B. Cohen, Z. Yang, S. P. Healey, R. E. Kennedy, and N. Gorelick. "A LandTrendr multispectral ensemble for forest disturbance detection." In: *Remote Sensing of Environment* 205.November 2017 (2018), pp. 131–140. ISSN: 00344257. DOI: 10.1016/j.rse.2017.11.015. URL: https://doi.org/10.1016/j.rse.2017.11.015.

[35] *Support Vector Machine: Kernel Trick; Mercer's Theorem*. URL: https://towardsdatascience.com/understanding-support-vector-machine-part-2-kernel-trick-mercers-theorem-e1e6848c6c4d (visited on 01/31/2020).

[36] *SVM: Maximum margin separating hyperplane — scikit-learn 0.22.1 documentation*. URL: https://scikit-learn.org/stable/auto_examples/svm/plot_separating_hyperplane.html (visited on 01/31/2020).

[37] E. Kim. "Everything You Wanted to Know about the Kernel Trick (But Were Too Afraid to Ask)." In: (2015), pp. 1–11. URL: https://www.eric-kim.net/eric-kim-net/posts/1/kernel_trick_blog_ekim_12_20_2017.pdf.

[38] *Top 4 advantages and disadvantages of Support Vector Machine or SVM*. URL: https://medium.com/@dhiraj8899/top-4-advantages-and-disadvantages-of-support-vector-machine-or-svm-a3c06a2b107 (visited on 01/31/2020).

[39] S. Maji, A. C. Berg, and J. Maliks. "Classification using intersection kernel support vector machines is efficient." In: *26th IEEE Conference on Computer Vision and Pattern Recognition, CVPR* (2008). DOI: 10.1109/CVPR.2008.4587630.

[40] J. R. R. Uijlings, K. E. A. van de Sande, T. Gevers, and A. Smeulders. "Selective Search for Object Recognition." In: *Proceedings - 13th IEEE International Conference on Automatic Face and Gesture Recognition, FG 2018* (2012), pp. 357–364. DOI: 10.1109/FG.2018.00058.

[41] H. D. Duong and D. T. Tinh. "An efficient method for vision-based fire detection using SVM classification." In: *2013 International Conference on Soft Computing and Pattern Recognition, SoCPaR 2013* (2013), pp. 190–195. DOI: 10.1109/SOCPAR.2013.7054125.

[42] T. Surasak, I. Takahiro, C. H. Cheng, C. E. Wang, and P. Y. Sheng. "Histogram of oriented gradients for human detection in video." In: *Proceedings of 2018 5th International Conference on Business and Industrial Research: Smart Technology for Next Generation of Information, Engineering, Business and Social Science, ICBIR 2018* (2018), pp. 172–176. DOI: 10.1109/ICBIR.2018.8391187.

[43] *Histogram of Oriented Gradients (HOG) Descriptor | Developer Reference for Intel® Integrated Performance Primitives*. URL: https://software.intel.com/en-us/ipp-dev-reference-histogram-of-oriented-gradients-hog-descriptor (visited on 01/31/2020).

[44] *The Artificial Neural Networks handbook: Part 1 - Coinmonks - Medium*. URL: https://medium.com/coinmonks/the-artificial-neural-networks-handbook-part-1-f9ceb0e376b4 (visited on 01/31/2020).

[45] A. Bhandare, M. Bhide, P. Gokhale, and R. Chandavarkar. "Applications of Convolutional Neural Networks." In: *International Journal of Computer Science and Information Technologies* 7.5 (2016), pp. 2206–2215. URL: http://ijcsit.com/docs/Volume7/vol7issue5/ijcsit20160705014.pdf.

[46] R. Girshick, J. Donahue, T. Darrell, and J. Malik. "Rich feature hierarchies for accurate object detection and semantic segmentation." In: *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition* (2014), pp. 580–587. ISSN: 10636919. DOI: 10.1109/CVPR.2014.81. arXiv: 1311.2524.

[47] A. Krizhevsky, I. Sutskever, and G. E. Hinton. "ImageNet classification with deep convolutional neural networks." In: *Communications of the ACM* 60.6 (June 2017), pp. 84–90. ISSN: 15577317. DOI: 10.1145/3065386.

[48] *ImageNet Large Scale Visual Recognition Competition 2012 (ILSVRC2012)*. URL: http://www.image-net.org/challenges/LSVRC/2012/ (visited on 01/31/2020).

[49] *Gradient descent explained - Learn ARCore - Fundamentals of Google ARCore [Book]*. URL: https://www.oreilly.com/library/view/learn-arcore-/9781788830409/e24a657a-a5c6-4ff2-b9ea-9418a7a5d24c.xhtml (visited on 02/06/2020).

[50] *Four Indonesian provinces, including Riau, declare disaster alerts for forest fires, SE Asia News & Top Stories - The Straits Times*. URL: https://www.straitstimes.com/asia/se-asia/four-indonesian-provinces-including-riau-declare-disaster-alerts-for-forest-fires (visited on 02/01/2020).

[51] *Blurring an Image | Apple Developer Documentation*. URL: https://developer.apple.com/documentation/accelerate/blurring_an_image (visited on 02/02/2020).

[52] *Fully Connected Layers in Convolutional Neural Networks: The Complete Guide - MissingLink.ai*. URL: https://missinglink.ai/guides/convolutional-neural-networks/fully-connected-layers-convolutional-neural-networks-complete-guide/ (visited on 02/02/2020).

[53] *A Beginner's Guide To Understanding Convolutional Neural Networks Part 1*. URL: https://www.kdnuggets.com/2016/09/beginners-guide-understanding-convolutional-neural-networks-part-1.html (visited on 02/02/2020).

[54] R. Girshick. "Fast R-CNN." In: *Proceedings of the IEEE International Conference on Computer Vision* 2015 Inter (2015), pp. 1440–1448. ISSN: 15505499. DOI: 10.1109/ICCV.2015.169. arXiv: 1504.08083.

[55] S. Ren, K. He, R. Girshick, and J. Sun. "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks." In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 39.6 (2017), pp. 1137–1149. ISSN: 01628828. DOI: 10.1109/TPAMI.2016.2577031. arXiv: 1506.01497.

[56] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. "You only look once: Unified, real-time object detection." In: *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition* 2016-Decem (2016), pp. 779–788. ISSN: 10636919. DOI: 10.1109/CVPR.2016.91. arXiv: 1506.02640.

[57] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C. Y. Fu, and A. C. Berg. "SSD: Single shot multibox detector." In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 9905 LNCS (2016), pp. 21–37. ISSN: 16113349. DOI: 10.1007/978-3-319-46448-0_2. arXiv: 1512.02325.

[58] K. He, G. Gkioxari, P. Dollar, and R. Girshick. "Mask R-CNN." In: *Proceedings of the IEEE International Conference on Computer Vision* 2017-Octob (2017), pp. 2980–2988. ISSN: 15505499. DOI: 10.1109/ICCV.2017.322. arXiv: 1703.06870.

[59] *GitHub - matterport/Mask$_R$CNN : MaskR−CNN for object detection and instance segmentation on Keras and TensorFlow*. URL: https://github.com/matterport/Mask_RCNN (visited on 11/09/2020).

[60] *TensorFlow*. URL: https://www.tensorflow.org/ (visited on 11/09/2020).

[61] *Keras: the Python deep learning API*. URL: https://keras.io/ (visited on 11/09/2020).

[62] *Review: FPN — Feature Pyramid Network (Object Detection) | by Sik-Ho Tsang | Towards Data Science*. URL: https://towardsdatascience.com/review-fpn-feature-pyramid-network-object-detection-262fc7482610 (visited on 11/09/2020).

[63] *A Brief History of CNNs in Image Segmentation: From R-CNN to Mask R-CNN | by Dhruv Parthasarathy | Athelas*. URL: https://blog.athelas.com/a-brief-history-of-cnns-in-image-segmentation-from-r-cnn-to-mask-r-cnn-34ea83205de4 (visited on 11/09/2020).

[64] *ROI pooling vs. ROI align. In computer vision there are many... | by Firiuza | Medium*. URL: https://medium.com/@Firiuza/roi-pooling-vs-roi-align-65293ab741db (visited on 11/09/2020).

[65] *Altum - MicaSense*. URL: https://micasense.com/altum/ (visited on 11/09/2020).

[66] *Agisoft Metashape*. URL: https://www.agisoft.com/ (visited on 11/09/2020).

[67] *Visual Geometry Group - University of Oxford*. URL: http://www.robots.ox.ac.uk/{~}vgg/software/via/ (visited on 11/09/2020).

[68] *maskrcnn_custom _tf_colab.ipynb - Colaboratory*. URL: https://colab.research.google.com/github/RomRoc/maskrcnn_train_tensorflow_colab/blob/master/maskrcnn_custom_tf_colab.ipynb (visited on 11/09/2020).

[69] K. He, X. Zhang, S. Ren, and J. Sun. "Deep residual learning for image recognition." In: *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. Vol. 2016-Decem. 2016, pp. 770–778. ISBN: 9781467388504. DOI: 10.1109/CVPR.2016.90. arXiv: 1512.03385. URL: http://image-net.org/challenges/LSVRC/2015/.

[70] *GitHub - aleju/imgaug: Image augmentation for machine learning experiments*. URL: https://github.com/aleju/imgaug (visited on 11/09/2020).

[71] *scikit-image: Image processing in Python — scikit-image*. URL: https://scikit-image.org/ (visited on 11/26/2020).

[72] *GitHub - pjreddie/darknet: Convolutional Neural Networks*. URL: https://github.com/pjreddie/darknet (visited on 11/10/2020).

[73] X. Zhou, D. Wang, and P. Krähenbühl. "Objects as Points." In: (2019). arXiv: 1904.07850. URL: http://arxiv.org/abs/1904.07850.

[74] *GitHub - tzutalin/labelImg: LabelImg is a graphical image annotation tool and label object bounding boxes in images*. URL: https://github.com/tzutalin/labelImg (visited on 11/09/2020).

[75] F. Araújo. "A Web-Based Dashboard for Improved Situational Awareness in Rekindle and Prescribed Fires." Master's thesis. Faculty of Sciences and Technology, NOVA University Lisbon, 2020.

[76] *Evaluation metrics for object detection and segmentation: mAP*. URL: https://kharshit.github.io/blog/2019/09/20/evaluation-metrics-for-object-detection-and-segmentation (visited on 11/07/2020).

[77] *Intersection over Union (IoU) for object detection - PyImageSearch*. URL: https://www.pyimagesearch.com/2016/11/07/intersection-over-union-iou-for-object-detection/ (visited on 11/07/2020).